Importing the required libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Importing the dataset

- **Method :** From Github account

In [2]:
```
# Setting the maximum column size to none so that we can view every single column

pd.set_option("display.max_columns", None)
data_input = pd.read_csv('https://raw.githubusercontent.com/yashu07/Nepal-Earthquake-Dataset-EDA/main/input_features.csv')
data_target = pd.read_csv('https://raw.githubusercontent.com/yashu07/Nepal-Earthquake-Dataset-EDA/main/target_values.csv')
```

In [3]:
```
print("The Shape of input dataset is {}".format(data_input.shape))
print("The Shape of target dataset is {}".format(data_target.shape))
```

```
The Shape of input dataset is (260601, 39)
The Shape of target dataset is (260601, 2)
```

- Input dataset consists of 260601 rows and 39 columns
- Target dataset consists of 260601 rows and 2 columns

In [4]:
```
# Merging the input and target dataset into single dataset which we will be using for further analysis

data = pd.merge(data_input, data_target, left_on = 'building_id', right_on= 'building_id')
```

In [5]:
```
print("The Shape of merged dataset is {}".format(data.shape))
```

```
The Shape of merged dataset is (260601, 40)
```

- After merging, the dataset consists of 260601 rows and 40 columns

In [6]:
```
# To print the first n rows of the data, in python default value is 5

data.head()
```

Out[6]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_condition	founda
0	802906	6	487	12198	2	30	6	5	t	
1	28830	8	900	2812	2	10	8	7	o	
2	94947	21	363	8973	2	10	5	5	t	
3	590882	22	418	10694	2	10	6	5	t	
4	201944	11	131	1488	3	30	8	9	t	

Understanding the dataset

The various aspects of building location and construction data consists of the following data columns:

- **building_id:** This column represents the Unique id of every buidling generated randomly
- **geo_level_1_id:** Largest geographic region in which building exists
- **geo_level_2_id:** Specific geographic region in which building exists
- **geo_level_3_id:** Most specific geographic sub-region in which building exists
- **count_floors_pre_eq:** Number of floors exists before earthquake
- **age:** This column shows the age of the building in years
- **area_percentage:** It shows the normalized area of the building footprint
- **height_percentage:** It shows the normalized height of the building footprint
- **land_surface_condition:** This columns helps in understating the surface condition of the land where the building was built
- **foundation_type:** It shows the foundation type used for construction of building
- **roof_type:** This column states the type of roof used while building

- **ground_floor_type:** It indicates the type of the ground floor
- **other_floor_type:** It indicates the type of construction used in higher floors other than the ground floor, excluding roof
- **position:** This column represents the position of the buidling
- **plan_configuration:** This column states the plan configuration of the structure
- **has_superstructure_adobe_mud:** Indicates if adobe mud was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_mud_mortar_stone:** Indicates if mud mortar stone was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_stone_flag:** Indicates if stone flag was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_cement_mortar_stone:** Indicates if cement mortar was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_mud_mortar_brick:** Indicates if mud mortar brick was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_cement_mortar_brick:** Indicates if cement mortar brick was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_timber:** Indicates if timber was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_bamboo:** Indicates if bamboo was used for construction of the structure. 1=Yes , 0 = No
- **has_superstructure_rc_non_engineered:** This column says whether the superstructure was rc non engineered
- **has_superstructure_rc_engineered:** This columns says whether the superstructure is rc engineered
- **has_superstructure_other:** Indicates if super structure was made of any other material
- **legal_ownership_status:** It show the legal status of the building
- **count_families:** Number of families living in the building
- **has_secondary_use:** Indicates if the building was used for secondary use or not. 1=Yes, 0=No
- **has_secondary_use_agriculture:** Indicates if building was used for agriculture purposes. 1=Yes, 0=No
- **has_secondary_use_hotel:** Indicates if building was used as a hotel. 1=Yes, 0=No
- **has_secondary_use_rental:** Indicates if building was used for rental purposes. 1=Yes, 0=No
- **has_secondary_use_institution:** Indicates if building was used as any institution. 1=Yes, 0=No
- **has_secondary_use_school:** Indicates if building was used as school. 1=Yes, 0=No
- **has_secondary_use_industry:** Indicates if building was used as a industry. 1=Yes, 0=No
- **has_secondary_use_health_post:** Indicates if building was used as a health post. 1=Yes, 0=No
- **has_secondary_use_gov_office:** Indicates if building was used as a government office. 1=Yes, 0=No
- **has_secondary_use_use_police:** Indicates if building was used as a police station. 1=Yes. 0=No
- **has_secondary_use_other:** Indicates if building was used for any other secondary purpose. 1=Yes, 0=No
- **damage_grade:** This column represents the damage levels on building that was hit by the earthquake as follows:

```
1 represents low damage
2 represents a medium amount of damage
3 represents almost complete destruction
```

In [7]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 260601 entries, 0 to 260600
Data columns (total 40 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   building_id                             260601 non-null  int64
 1   geo_level_1_id                          260601 non-null  int64
 2   geo_level_2_id                          260601 non-null  int64
 3   geo_level_3_id                          260601 non-null  int64
 4   count_floors_pre_eq                     260601 non-null  int64
 5   age                                     260601 non-null  int64
 6   area_percentage                         260601 non-null  int64
 7   height_percentage                       260601 non-null  int64
 8   land_surface_condition                  260601 non-null  object
 9   foundation_type                         260601 non-null  object
 10  roof_type                               260601 non-null  object
 11  ground_floor_type                       260601 non-null  object
 12  other_floor_type                        260601 non-null  object
 13  position                                260601 non-null  object
 14  plan_configuration                      260601 non-null  object
 15  has_superstructure_adobe_mud            260601 non-null  int64
 16  has_superstructure_mud_mortar_stone     260601 non-null  int64
 17  has_superstructure_stone_flag           260601 non-null  int64
 18  has_superstructure_cement_mortar_stone  260601 non-null  int64
 19  has_superstructure_mud_mortar_brick     260601 non-null  int64
 20  has_superstructure_cement_mortar_brick  260601 non-null  int64
 21  has_superstructure_timber               260601 non-null  int64
 22  has_superstructure_bamboo               260601 non-null  int64
 23  has_superstructure_rc_non_engineered    260601 non-null  int64
 24  has_superstructure_rc_engineered        260601 non-null  int64
 25  has_superstructure_other                260601 non-null  int64
 26  legal_ownership_status                  260601 non-null  object
 27  count_families                          260601 non-null  int64
 28  has_secondary_use                       260601 non-null  int64
 29  has_secondary_use_agriculture           260601 non-null  int64
 30  has_secondary_use_hotel                 260601 non-null  int64
 31  has_secondary_use_rental                260601 non-null  int64
 32  has_secondary_use_institution           260601 non-null  int64
 33  has_secondary_use_school                260601 non-null  int64
 34  has_secondary_use_industry              260601 non-null  int64
 35  has_secondary_use_health_post           260601 non-null  int64
 36  has_secondary_use_gov_office            260601 non-null  int64
 37  has_secondary_use_use_police            260601 non-null  int64
 38  has_secondary_use_other                 260601 non-null  int64
```

```
 39  damage_grade                            260601 non-null  int64
dtypes: int64(32), object(8)
memory usage: 81.5+ MB
```

In [8]:

```
# To check for any duplicate of the data

data.duplicated().sum()
```

Out[8]:

```
0
```

- There are no duplicate rows in the dataset.

In [9]:

```
# To check for any null value in the dataset


data.isnull().sum()
```

Out[9]:

```
building_id                             0
geo_level_1_id                          0
geo_level_2_id                          0
geo_level_3_id                          0
count_floors_pre_eq                     0
age                                     0
area_percentage                         0
height_percentage                       0
land_surface_condition                  0
foundation_type                         0
roof_type                               0
ground_floor_type                       0
other_floor_type                        0
position                                0
plan_configuration                      0
has_superstructure_adobe_mud            0
has_superstructure_mud_mortar_stone     0
has_superstructure_stone_flag           0
has_superstructure_cement_mortar_stone  0
has_superstructure_mud_mortar_brick     0
has_superstructure_cement_mortar_brick  0
has_superstructure_timber               0
has_superstructure_bamboo               0
has_superstructure_rc_non_engineered    0
has_superstructure_rc_engineered        0
has_superstructure_other                0
```

```
legal_ownership_status                  0
count_families                          0
has_secondary_use                       0
has_secondary_use_agriculture           0
has_secondary_use_hotel                 0
has_secondary_use_rental                0
has_secondary_use_institution           0
has_secondary_use_school                0
has_secondary_use_industry              0
has_secondary_use_health_post           0
has_secondary_use_gov_office            0
has_secondary_use_use_police            0
has_secondary_use_other                 0
damage_grade                            0
dtype: int64
```

- Moreover, using dataframe.isnull().sum(), it is evident that dataset does not contain any Null/NaN value

In [10]:
```
data.nunique(axis=0)
```

Out[10]:
```
building_id                            260601
geo_level_1_id                             31
geo_level_2_id                           1414
geo_level_3_id                          11595
count_floors_pre_eq                         9
age                                        42
area_percentage                            84
height_percentage                          27
land_surface_condition                      3
foundation_type                             5
roof_type                                   3
ground_floor_type                           5
other_floor_type                            4
position                                    4
plan_configuration                         10
has_superstructure_adobe_mud                2
has_superstructure_mud_mortar_stone         2
has_superstructure_stone_flag               2
has_superstructure_cement_mortar_stone      2
has_superstructure_mud_mortar_brick         2
has_superstructure_cement_mortar_brick      2
has_superstructure_timber                   2
has_superstructure_bamboo                   2
has_superstructure_rc_non_engineered        2
has_superstructure_rc_engineered            2
```

```
has_superstructure_other                 2
legal_ownership_status                   4
count_families                          10
has_secondary_use                        2
has_secondary_use_agriculture            2
has_secondary_use_hotel                  2
has_secondary_use_rental                 2
has_secondary_use_institution            2
has_secondary_use_school                 2
has_secondary_use_industry               2
has_secondary_use_health_post            2
has_secondary_use_gov_office             2
has_secondary_use_use_police             2
has_secondary_use_other                  2
damage_grade                             3
dtype: int64
```

- The above code block illustrates the number of different values in each column.

In [11]:
```
data.columns
```

Out[11]:
```
Index(['building_id', 'geo_level_1_id', 'geo_level_2_id', 'geo_level_3_id',
       'count_floors_pre_eq', 'age', 'area_percentage', 'height_percentage',
       'land_surface_condition', 'foundation_type', 'roof_type',
       'ground_floor_type', 'other_floor_type', 'position',
       'plan_configuration', 'has_superstructure_adobe_mud',
       'has_superstructure_mud_mortar_stone', 'has_superstructure_stone_flag',
       'has_superstructure_cement_mortar_stone',
       'has_superstructure_mud_mortar_brick',
       'has_superstructure_cement_mortar_brick', 'has_superstructure_timber',
       'has_superstructure_bamboo', 'has_superstructure_rc_non_engineered',
       'has_superstructure_rc_engineered', 'has_superstructure_other',
       'legal_ownership_status', 'count_families', 'has_secondary_use',
       'has_secondary_use_agriculture', 'has_secondary_use_hotel',
       'has_secondary_use_rental', 'has_secondary_use_institution',
       'has_secondary_use_school', 'has_secondary_use_industry',
       'has_secondary_use_health_post', 'has_secondary_use_gov_office',
       'has_secondary_use_use_police', 'has_secondary_use_other',
       'damage_grade'],
      dtype='object')
```

- All the columns from the dataset.

In [12]:
```

```

```
# To print the first 8 rows of the dataset

data.head(8)
```

Out[12]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_condition	founda
0	802906	6	487	12198	2	30	6	5	t	
1	28830	8	900	2812	2	10	8	7	o	
2	94947	21	363	8973	2	10	5	5	t	
3	590882	22	418	10694	2	10	6	5	t	
4	201944	11	131	1488	3	30	8	9	t	
5	333020	8	558	6089	2	10	9	5	t	
6	728451	9	475	12066	2	25	3	4	n	
7	475515	20	323	12236	2	0	8	6	t	

In [13]:

```
# To print the last 8 rows of the dataset

data.tail(8)
```

Out[13]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_condition	f
260593	226421	8	767	8613	2	5	13	5	t	
260594	159555	27	181	1537	6	0	13	12	t	
260595	827012	8	268	4718	2	20	8	5	t	
260596	688636	25	1335	1621	1	55	6	3	n	
260597	669485	17	715	2060	2	0	6	5	t	
260598	602512	17	51	8163	3	55	6	7	t	
260599	151409	26	39	1851	2	10	14	6	t	
260600	747594	21	9	9101	3	10	7	6	n	

In [14]:
```
data.describe()
```

Out[14]:

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	has_superstruc
count	2.606010e+05	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	
mean	5.256755e+05	13.900353	701.074685	6257.876148	2.129723	26.535029	8.018051	5.434365	
std	3.045450e+05	8.033617	412.710734	3646.369645	0.727665	73.565937	4.392231	1.918418	
min	4.000000e+00	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	2.000000	
25%	2.611900e+05	7.000000	350.000000	3073.000000	2.000000	10.000000	5.000000	4.000000	
50%	5.257570e+05	12.000000	702.000000	6270.000000	2.000000	15.000000	7.000000	5.000000	
75%	7.897620e+05	21.000000	1050.000000	9412.000000	2.000000	30.000000	9.000000	6.000000	
max	1.052934e+06	30.000000	1427.000000	12567.000000	9.000000	995.000000	100.000000	32.000000	

- The above method is used for calculating the statistical data, namely, mean, count, five number summary and standard deviation of the column having int/float datatypes.

In [15]:
```
# To calculate the additional statistical data such as Unique, Frequency and most used values in the column containing categorical

data.describe(include = 'all')
```

Out[15]:

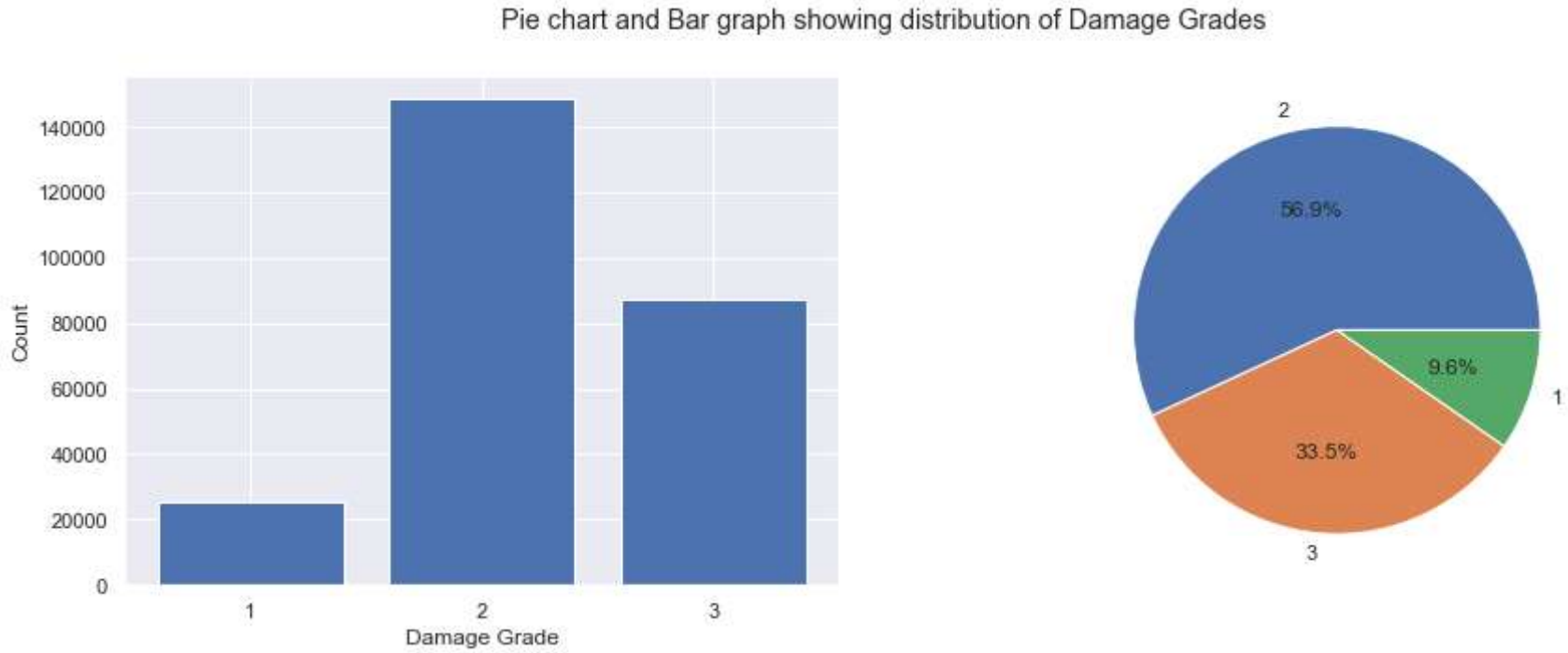
	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_
count	2.606010e+05	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	260601.000000	
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	5.256755e+05	13.900353	701.074685	6257.876148	2.129723	26.535029	8.018051	5.434365	
std	3.045450e+05	8.033617	412.710734	3646.369645	0.727665	73.565937	4.392231	1.918418	

	building_id	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_
min	4.000000e+00	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	2.000000	
25%	2.611900e+05	7.000000	350.000000	3073.000000	2.000000	10.000000	5.000000	4.000000	
50%	5.257570e+05	12.000000	702.000000	6270.000000	2.000000	15.000000	7.000000	5.000000	
75%	7.897620e+05	21.000000	1050.000000	9412.000000	2.000000	30.000000	9.000000	6.000000	
max	1.052934e+06	30.000000	1427.000000	12567.000000	9.000000	995.000000	100.000000	32.000000	

Damage Grade Distribution

In [16]:
```
# Creating new directory

damage_grade_dict =  dict(data.damage_grade.value_counts())
```

In [17]:
```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))

fig.suptitle('Pie chart and Bar graph showing distribution of Damage Grades')
axs[0].bar(damage_grade_dict.keys(),damage_grade_dict.values())
axs[0].set_xlabel('Damage Grade')
axs[0].set_ylabel('Count')
axs[0].set_xticks(range(1,4))
axs[1].pie(damage_grade_dict.values(),labels = damage_grade_dict.keys(), autopct='%1.1f%%')
plt.show()
```



- It is clear that the dataset has unbalanced distribution since 56.9% of the total entries have damage grade 2. Hence, the damage grade 2 is dominating.

Correlation

In [18]:
```
data_correlation = data.drop('damage_grade',axis=1)
```

- Droppping the damage garde and copying the dataset into new variable to find the correlation of each feature with damage grade.

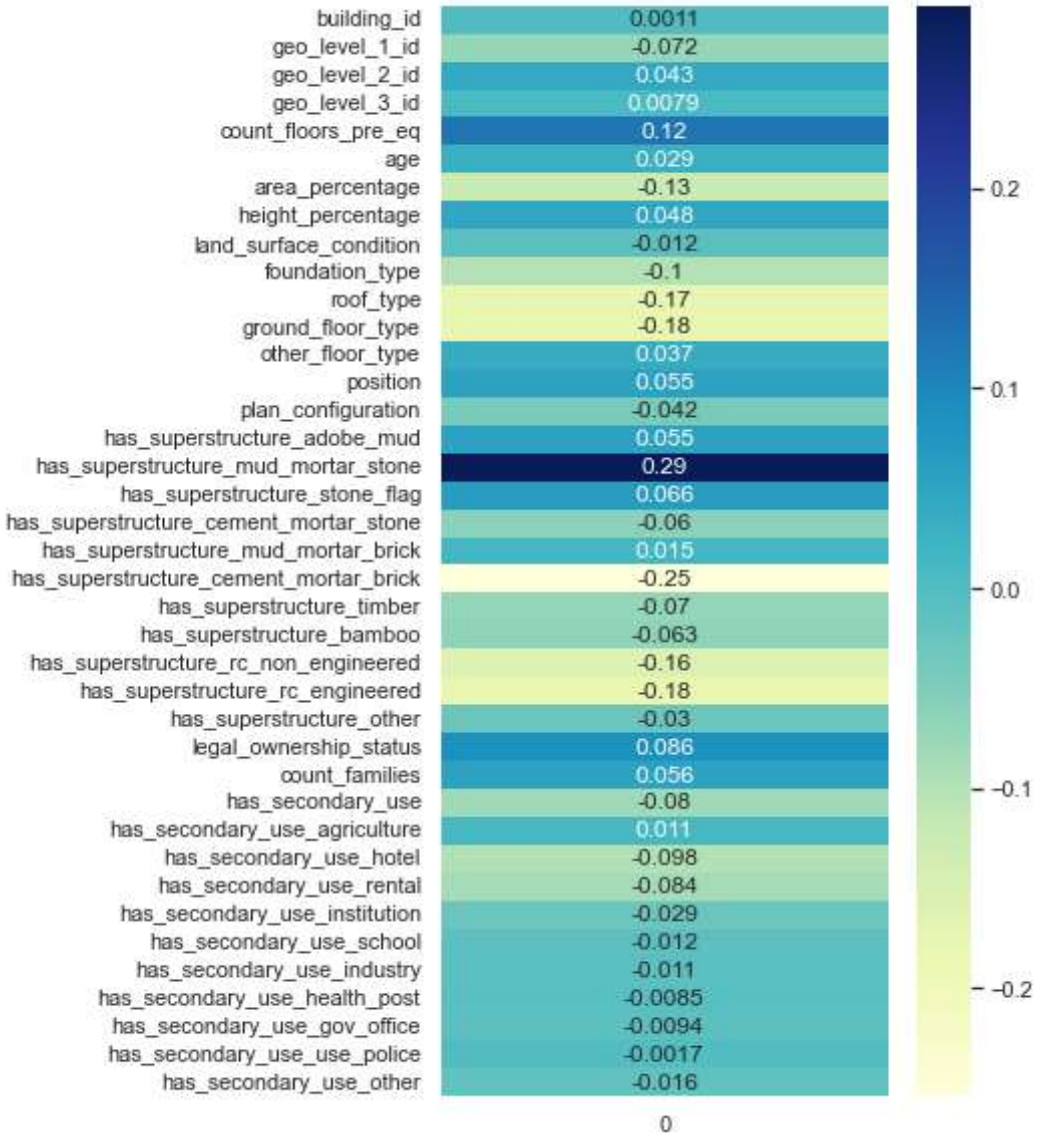
In [19]:
```
data_correlation['land_surface_condition'] =data_correlation['land_surface_condition'].astype('category').cat.codes
data_correlation['foundation_type'] =data_correlation['foundation_type'].astype('category').cat.codes
data_correlation['roof_type'] =data_correlation['roof_type'].astype('category').cat.codes
data_correlation['ground_floor_type'] =data_correlation['ground_floor_type'].astype('category').cat.codes
data_correlation['other_floor_type'] =data_correlation['other_floor_type'].astype('category').cat.codes
data_correlation['position'] =data_correlation['position'].astype('category').cat.codes
```

```
data_correlation['plan_configuration'] =data_correlation['plan_configuration'].astype('category').cat.codes
data_correlation['legal_ownership_status'] =data_correlation['legal_ownership_status'].astype('category').cat.codes
```

- We have converted the features having categorical variables into numeric data so that we can use catrgorical variables in finding the correlation.

In [20]:

```
correlation =  data_correlation.corrwith(data.damage_grade)
fig, ax = plt.subplots(figsize=(5,10))
sns.heatmap(pd.DataFrame(correlation),annot = True,cmap="YlGnBu")
plt.show()
```



- The above block illustrates the heatmap containing correlation values with the damage grade.
- It is observed that there is mild positive & negative correlation between features and damage grade.

Feature Analysis, Preprocessing and Cleaning

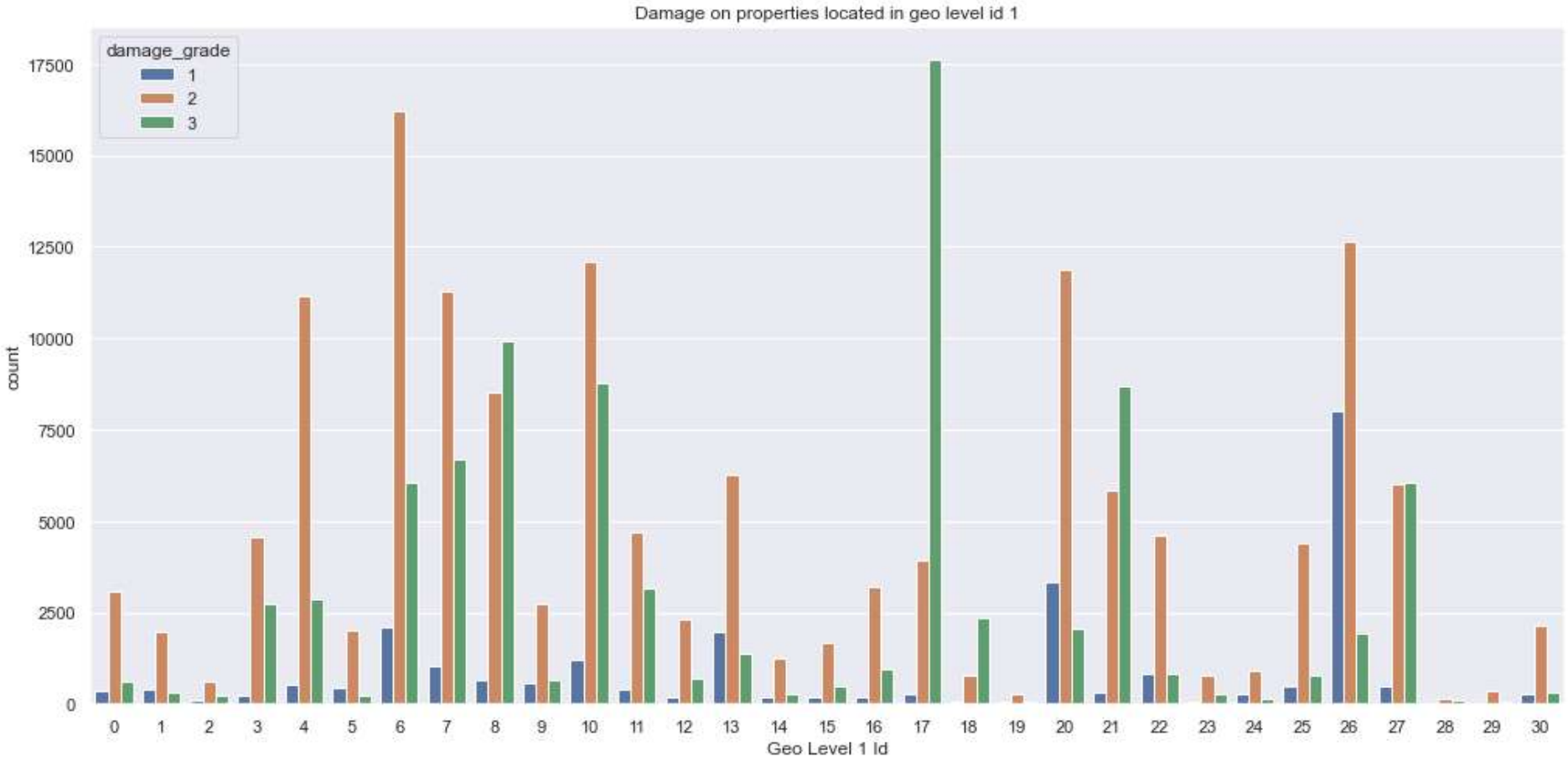
Dropping the building_id column

In [21]:
```
data = data.drop('building_id',axis=1)
```

- Here, we have dropped the column 'building_id' as it is not required for the further analysis.

Geo Level ID 1

In [22]:
```
plt.figure(figsize=(17,8))
plt.title('Damage on properties located in geo level id 1')
sns.countplot(x='geo_level_1_id',data=data,hue='damage_grade')
plt.xlabel('Geo Level 1 Id')
plt.show()
```

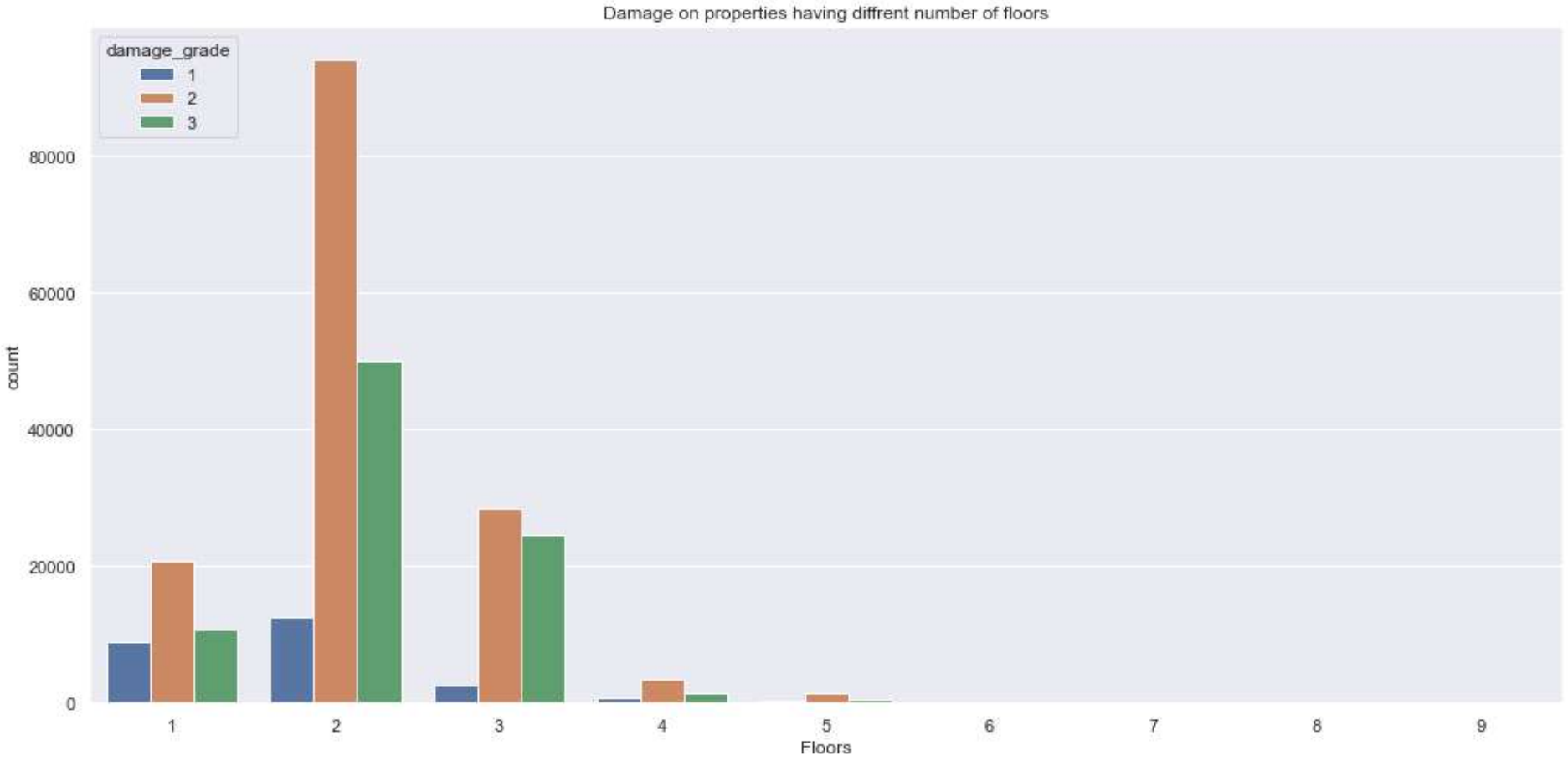


- Certainly, from the countplot of geo level id 1 with the damage grade, it is clear that the buildings with ID - 8, 17, 21, 27 have recoreded more of level 3 damage than the level 2 damage. However, building id 17 in geo level 1 has the maximum amount of level 3 damage.

Count Floors

In [23]:

```
plt.figure(figsize=(17,8))
plt.title('Damage on properties having diffrent number of floors')
sns.countplot(x='count_floors_pre_eq',data=data,hue='damage_grade')
plt.xlabel('Floors')
plt.show()
```

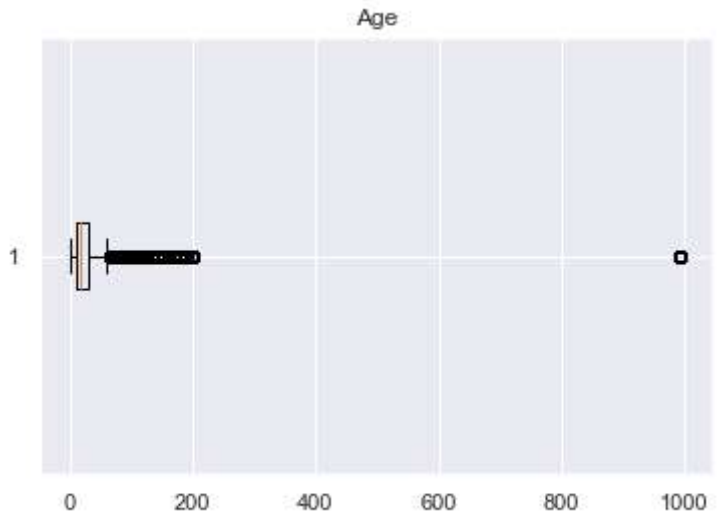


- The damage grade was correlated with the number of floors in the building. Unfortunately, no specific trend/special case was observed.

Age

In [24]:

```
plt.boxplot(data.age,vert=False)
plt.title("Age")
plt.show()
```



- The boxplot of the age of the building exemplifies that the age is positively skewed. Moreover, it is observed that there is value near 1000 away from most of the values near 200. We can consider that as an outlier.

In [25]:
```
data.age.value_counts()
```

Out[25]:
```
10     38896
15     36010
5      33697
20     32182
0      26041
25     24366
30     18028
35     10710
40     10559
50      7257
45      4711
60      3612
80      3055
55      2033
70      1975
995     1390
100     1364
65      1123
90      1085
85       847
75       512
```

```
95       414
120      180
150      142
200      106
110      100
105       89
125       37
115       21
130        9
140        9
180        7
160        6
170        6
175        5
135        5
190        3
145        3
195        2
165        2
155        1
185        1
Name: age, dtype: int64
```

- By getting the value counts of age, We found there are 1390 buildings which are 995 old.

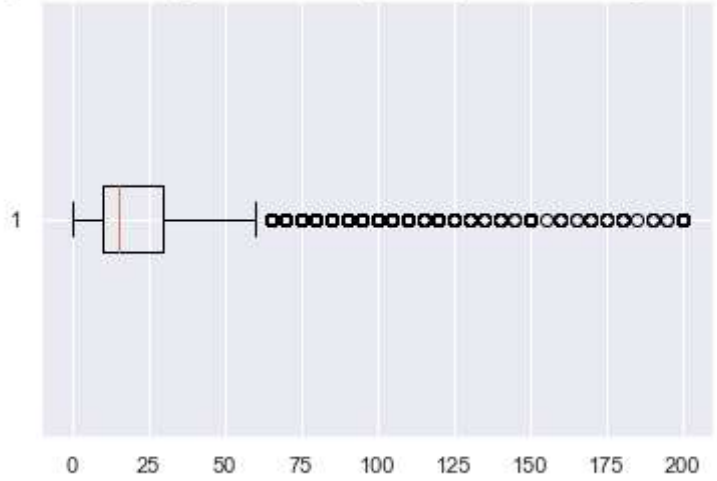
In [26]:

```
# Removing the outlier value i.e 995
data = data[data.age<995]

# Applying one of the Data wrangling technique to reset the index and getting a proper data after dropping inital index
data = data.reset_index()
data = data.drop('index',axis=1)
```

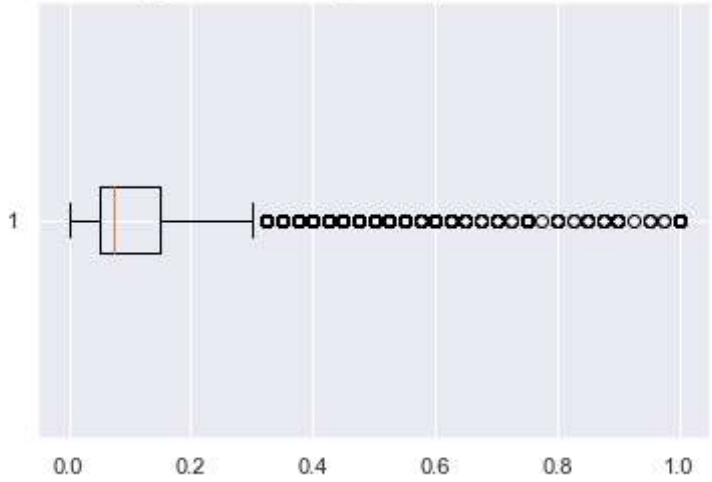
In [27]:

```
plt.boxplot(data.age,vert=False)
plt.title("Age after removing the outlier value(i.e. 995) but before taking natural log")
plt.show()

# Min Max Normalization of age
data['minmax_normalize_age'] =  (data.age - data.age.min())/(data.age.max() -  data.age.min())

plt.boxplot(data.minmax_normalize_age,vert=False)
plt.title("Age after removing the outlier value(i.e. 995) and after Min Max Normalization")
plt.show()
```

Age after removing the outlier value(i.e. 995) but before taking natural log



Age after removing the outlier value(i.e. 995) and after Min Max Normalization

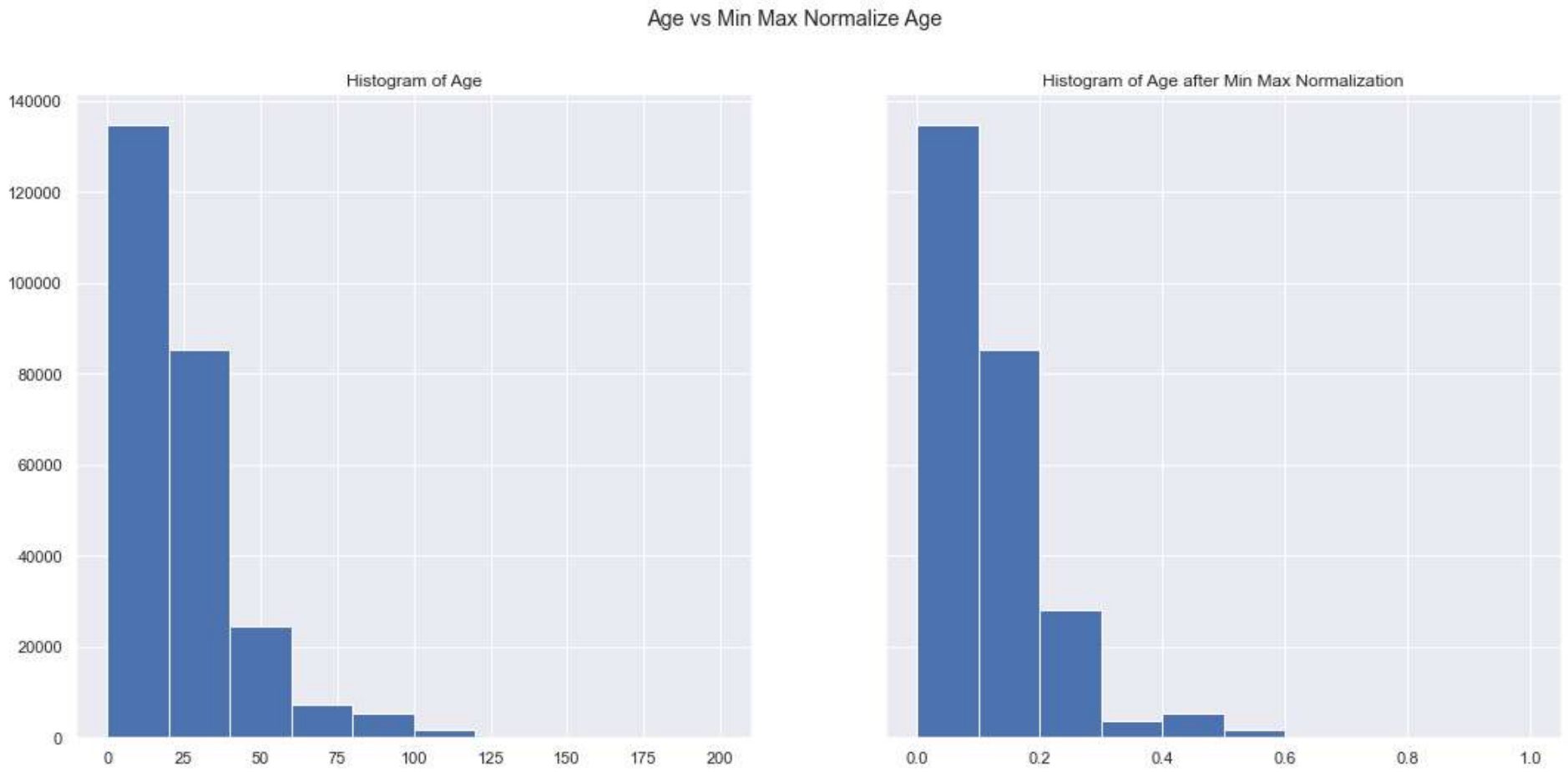


- There was no change in the age boxplot after normalization. In the boxplot of Min Max Normalization, the x-axis value changes from 0 to 1.

In [28]:

```
fig, axs = plt.subplots(1, 2, figsize=(18, 8), sharey = True)
fig.suptitle('Age vs Min Max Normalize Age')
axs[0].hist(data.age)
axs[0].set_title('Histogram of Age')

axs[1].hist(data.minmax_normalize_age)
```

```
axs[1].set_title('Histogram of Age after Min Max Normalization')

plt.show()
```

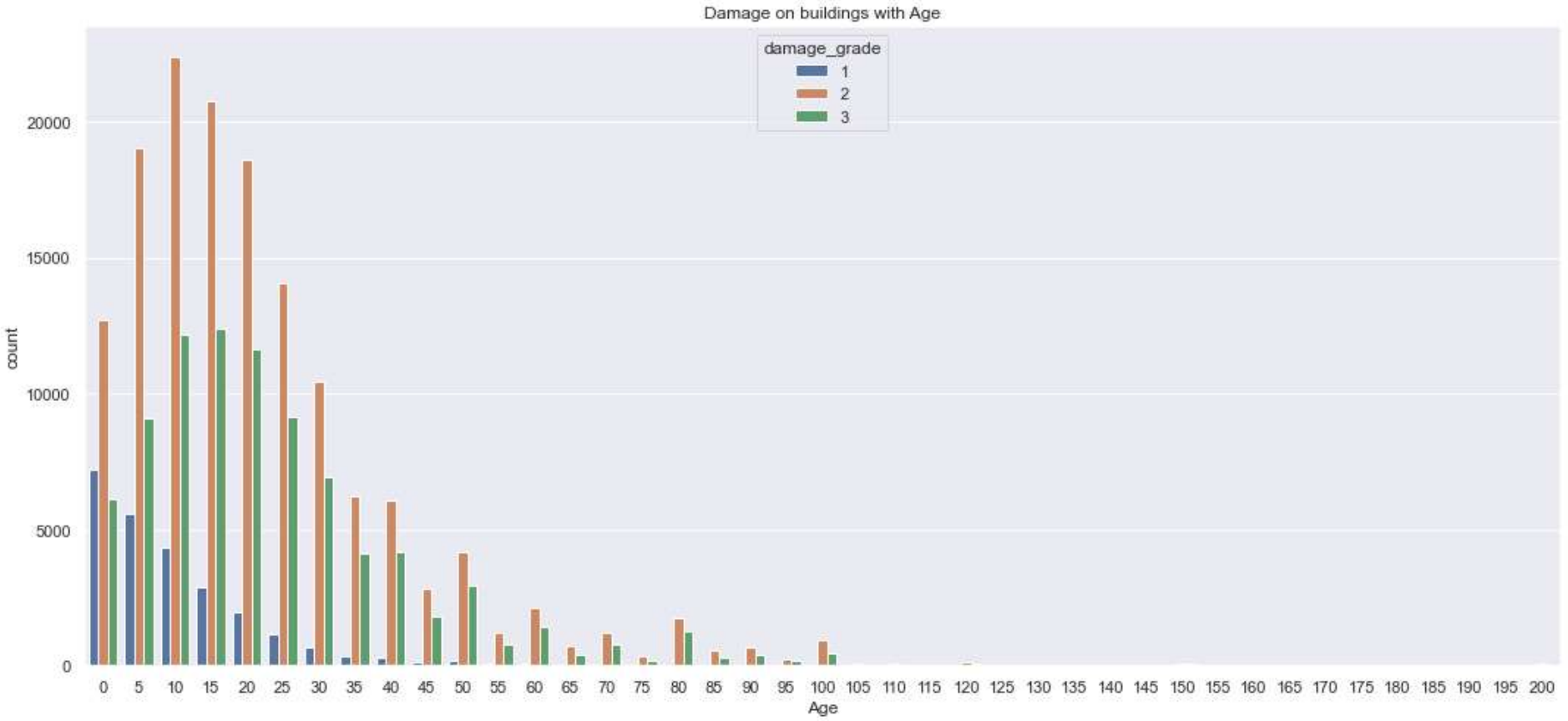


- It is visible from the above histogram that the age is positively skewed. Inorder to achieve normal distribution, logarithmic transformation can be used. However, the given dataset consists of over 26000 newly built buildings (i.e age = 0) and log(0) is undefined, therefore, the use of logarithmic transformation is not possible.

In [29]:

```
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with Age')
sns.countplot(x='age',data=data,hue='damage_grade')
```

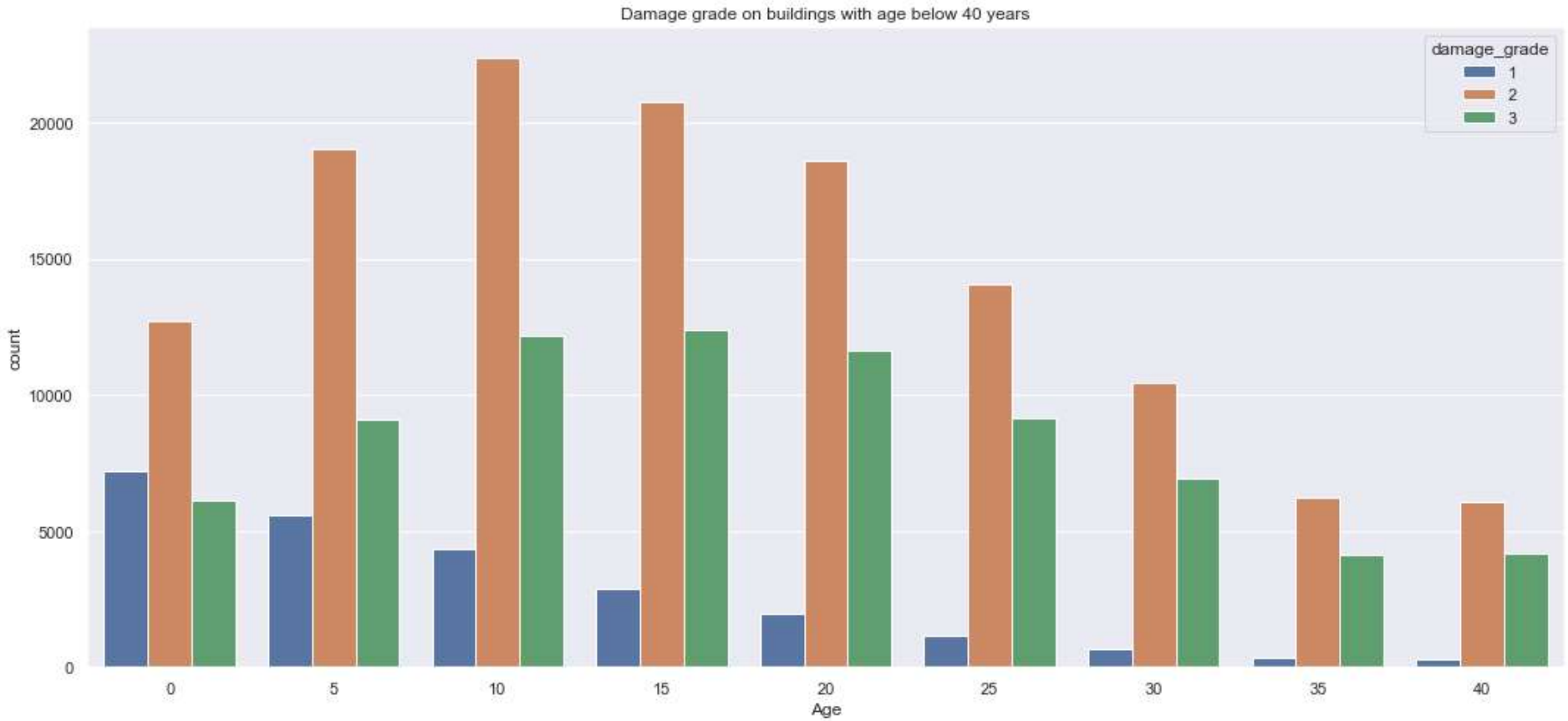
```
plt.xlabel('Age')
plt.show()
```



- In the above countplot of building's age with a particular damage grade, it is observed that as the building's age increases, the number of buildings with damage grade 1 decreases.

In [30]:

```
plt.figure(figsize=(18,8))
plt.title('Damage grade on buildings with age below 40 years')
buildings_age_less_than_40 =data[data['age']<=40]
sns.countplot(x='age',data=buildings_age_less_than_40,hue='damage_grade')
plt.xlabel('Age')
plt.show()
```



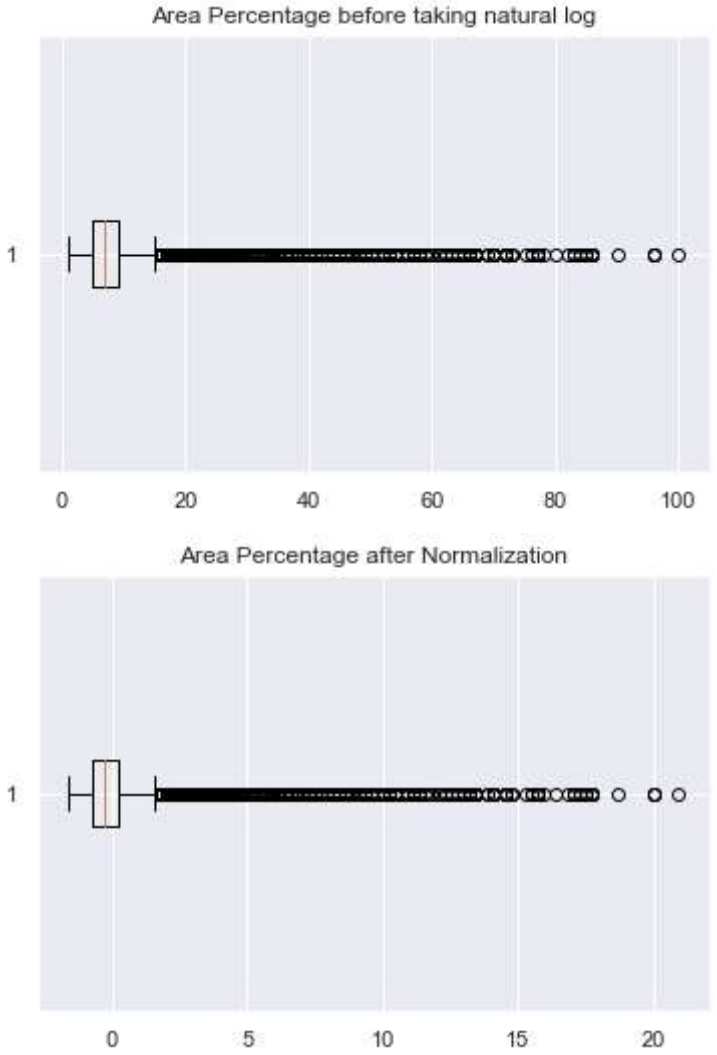
- In the above countplot of building's age less than 40 years old with a particular damage grade, it is observed that as the building's age increases, the number of buildings with damage grade 1 decreases.

Area Percentage

In [31]:

```
plt.boxplot(data.area_percentage,vert=False)
plt.title("Area Percentage before taking natural log")
plt.show()

# Performing Normalization on area percentage.
data['normalize_area_percentage']= (data.area_percentage - data.area_percentage.mean())/data.area_percentage.std()
```

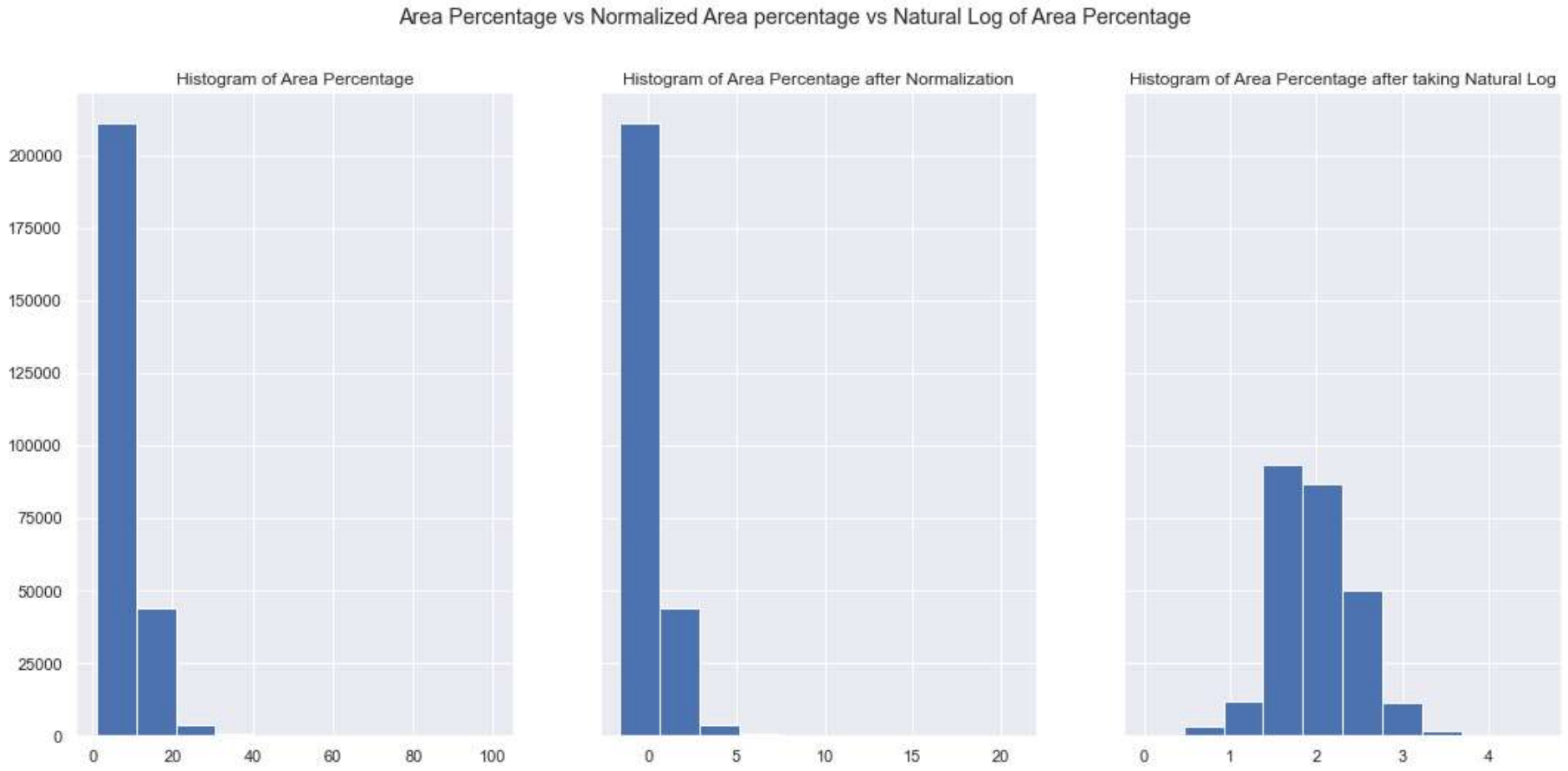
```
plt.boxplot(data.normalize_area_percentage,vert=False)
plt.title("Area Percentage after Normalization")
plt.show()
```



- Area percentage is also found to be positively skewed.

In [32]:
```
# To check the effect on area percentage after normalization
```

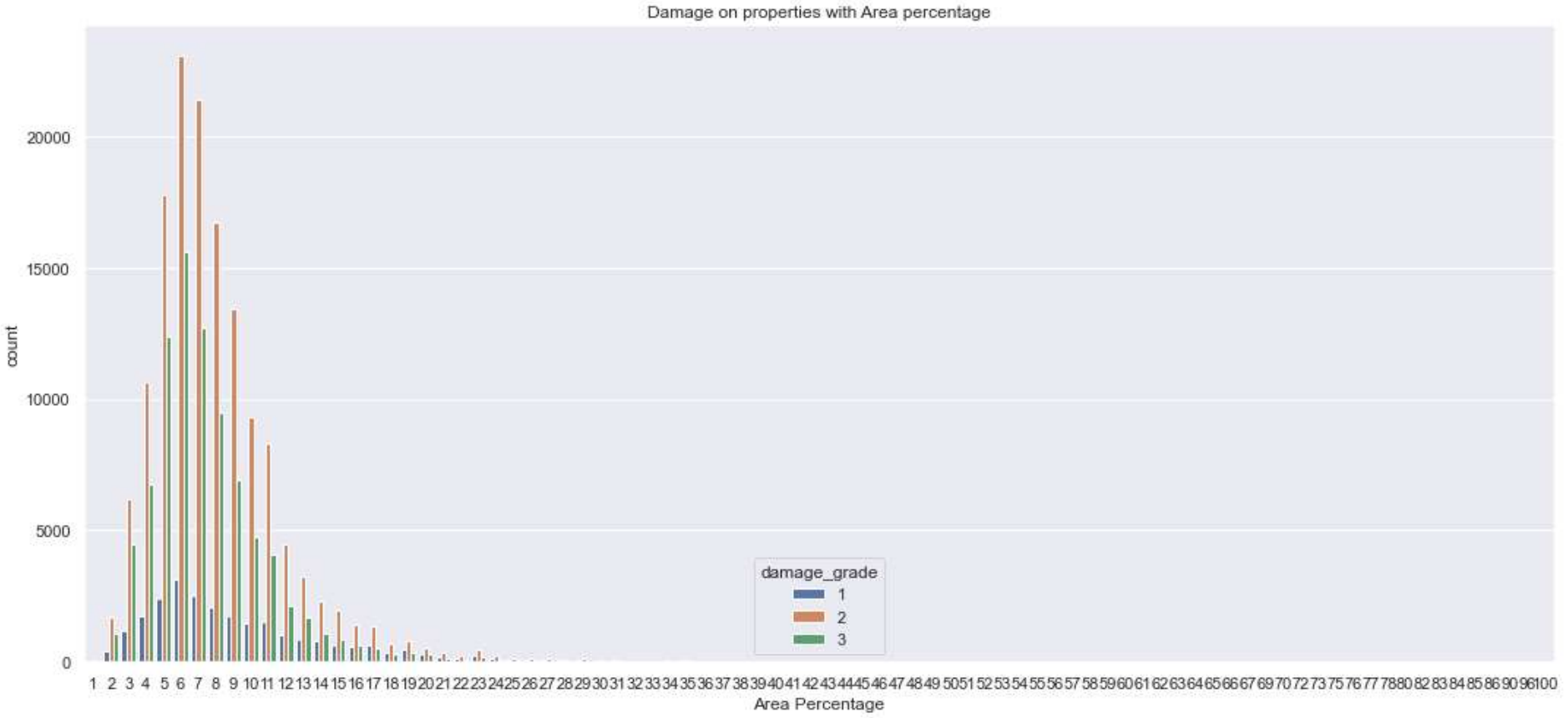
```
fig, axs = plt.subplots(1, 3, figsize=(18, 8), sharey = True)
fig.suptitle('Area Percentage vs Normalized Area percentage vs Natural Log of Area Percentage')
axs[0].hist(data.area_percentage)
axs[0].set_title('Histogram of Area Percentage')

axs[1].hist(data.normalize_area_percentage)
axs[1].set_title('Histogram of Area Percentage after Normalization')

axs[2].hist(np.log(data.area_percentage))
axs[2].set_title('Histogram of Area Percentage after taking Natural Log')

plt.show()
```



- From the three different histograms above, it is evident that the normalization does not affect the skewness of area percentage but the log transformation of area percentage leads to nearly symmetrical distribution.

In [33]:

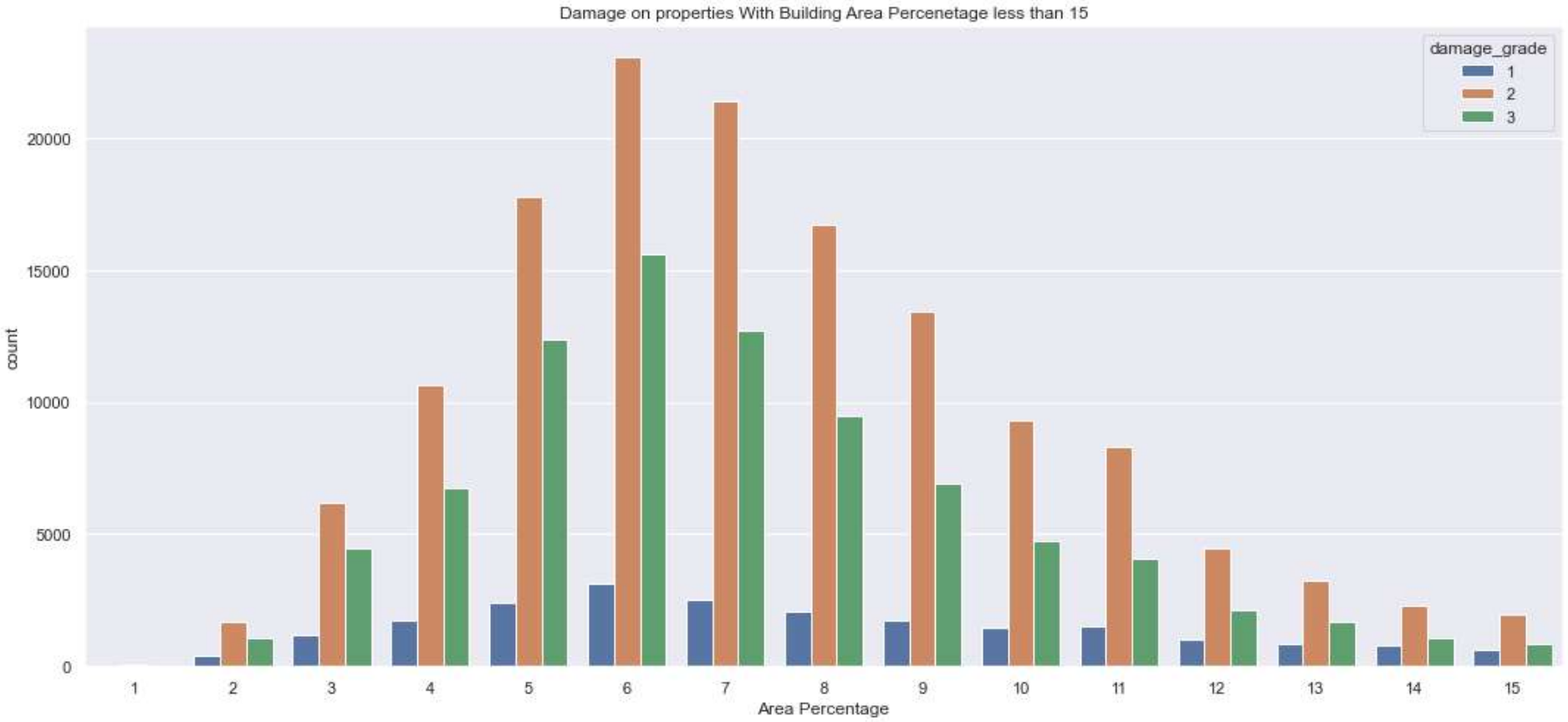
```
plt.figure(figsize=(18,8))
plt.title('Damage on properties with Area percentage')
sns.countplot(x='area_percentage',data=data,hue='damage_grade')
plt.xlabel('Area Percentage')
plt.show()
```



In [34]:

```
plt.figure(figsize=(18,8))
plt.title('Damage on properties With Building Area Percenetage less than 15')
buildings_area_percentage_less_than_15 =data[data['area_percentage']<=15]
sns.countplot(x='area_percentage',data=buildings_area_percentage_less_than_15,hue='damage_grade')
```

```
plt.xlabel('Area Percentage')
plt.show()
```



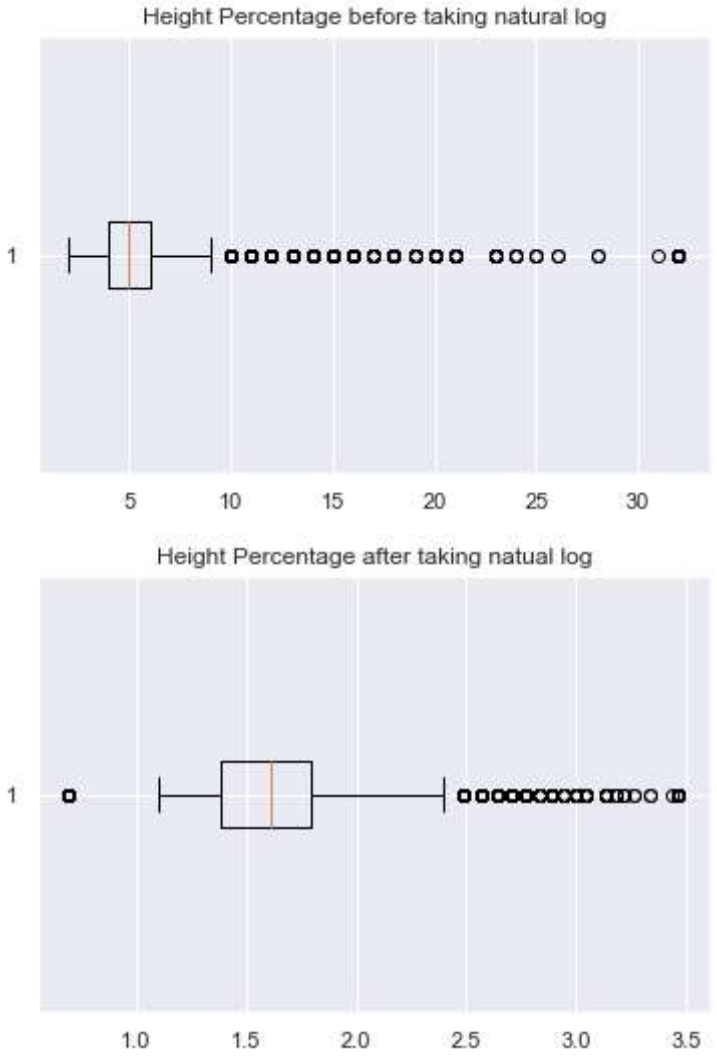
- From the countplots of Area percentage with damage grades, unfortunately no particular trend was observed.

Height Percentage

In [35]:

```
plt.boxplot(data.height_percentage,vert=False)
plt.title("Height Percentage before taking natural log")
plt.show()

data['log_height_percentage']=np.log(data.height_percentage)
```

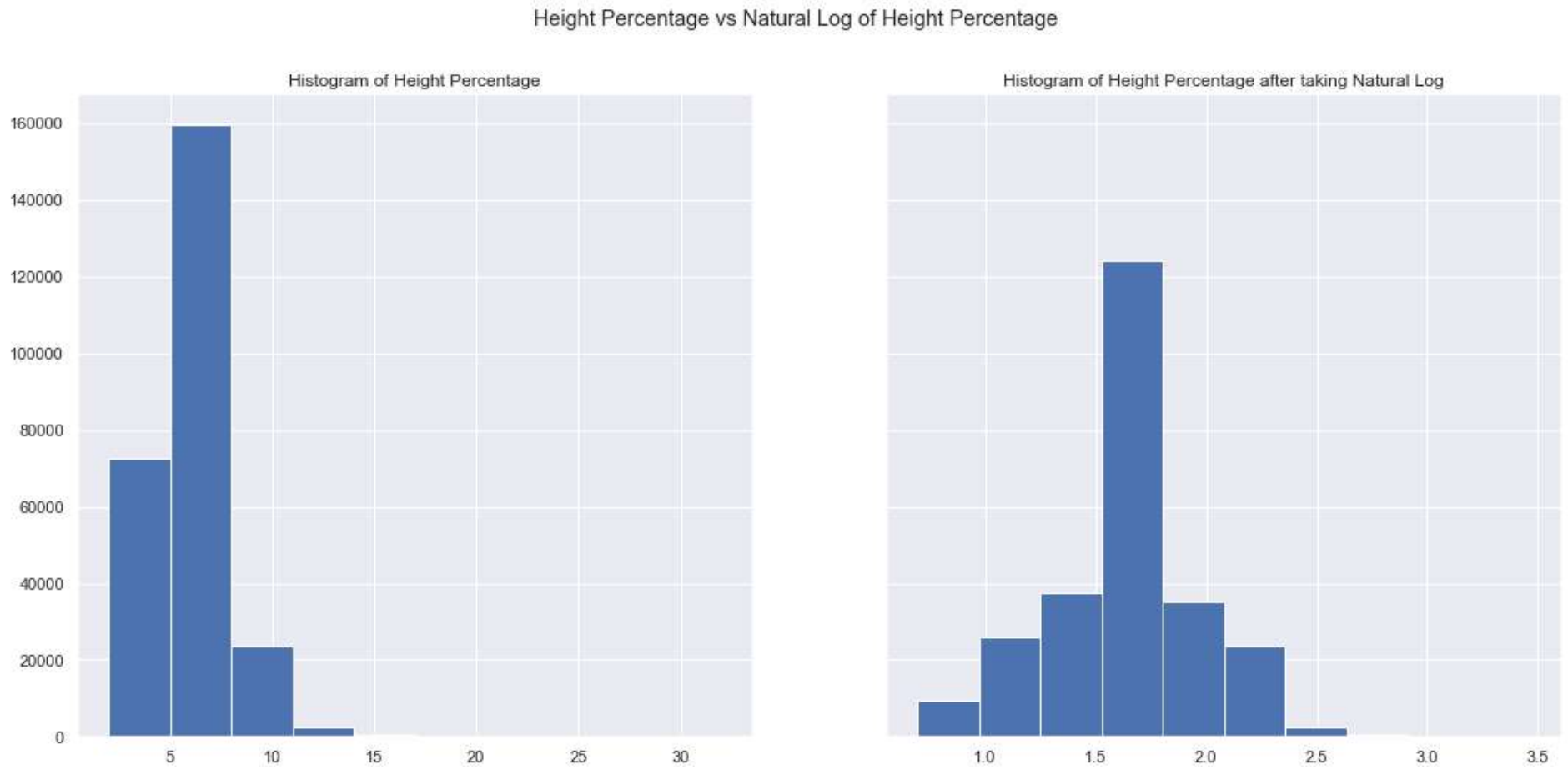
```
plt.boxplot(data.log_height_percentage,vert=False)
plt.title("Height Percentage after taking natual log")
plt.show()
```



In [36]:

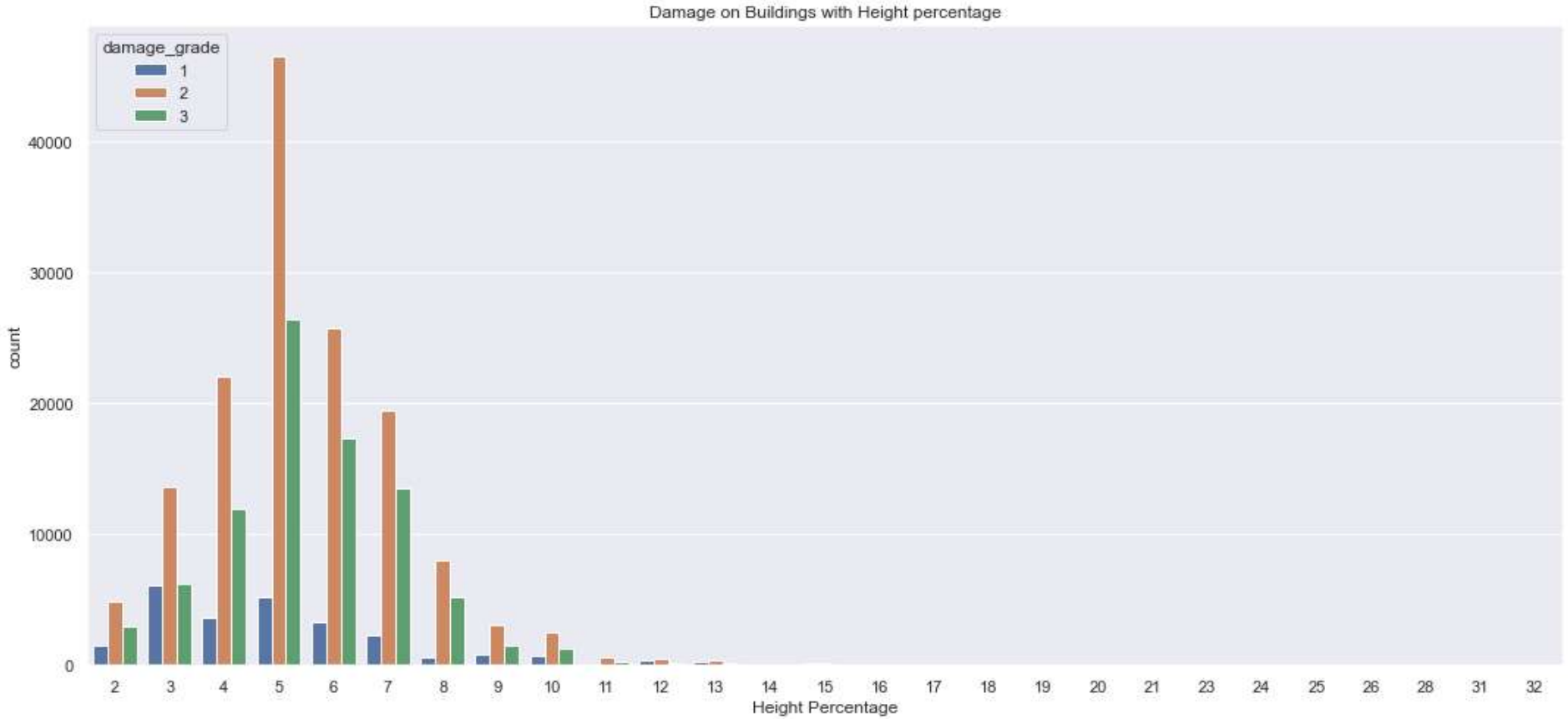
```
fig, axs = plt.subplots(1, 2, figsize=(18, 8), sharey = True)
fig.suptitle('Height Percentage vs Natural Log of Height Percentage')
axs[0].hist(data.height_percentage)
axs[0].set_title('Histogram of Height Percentage')
```

```
axs[1].hist(data.log_height_percentage)
axs[1].set_title('Histogram of Height Percentage after taking Natural Log')

plt.show()
```



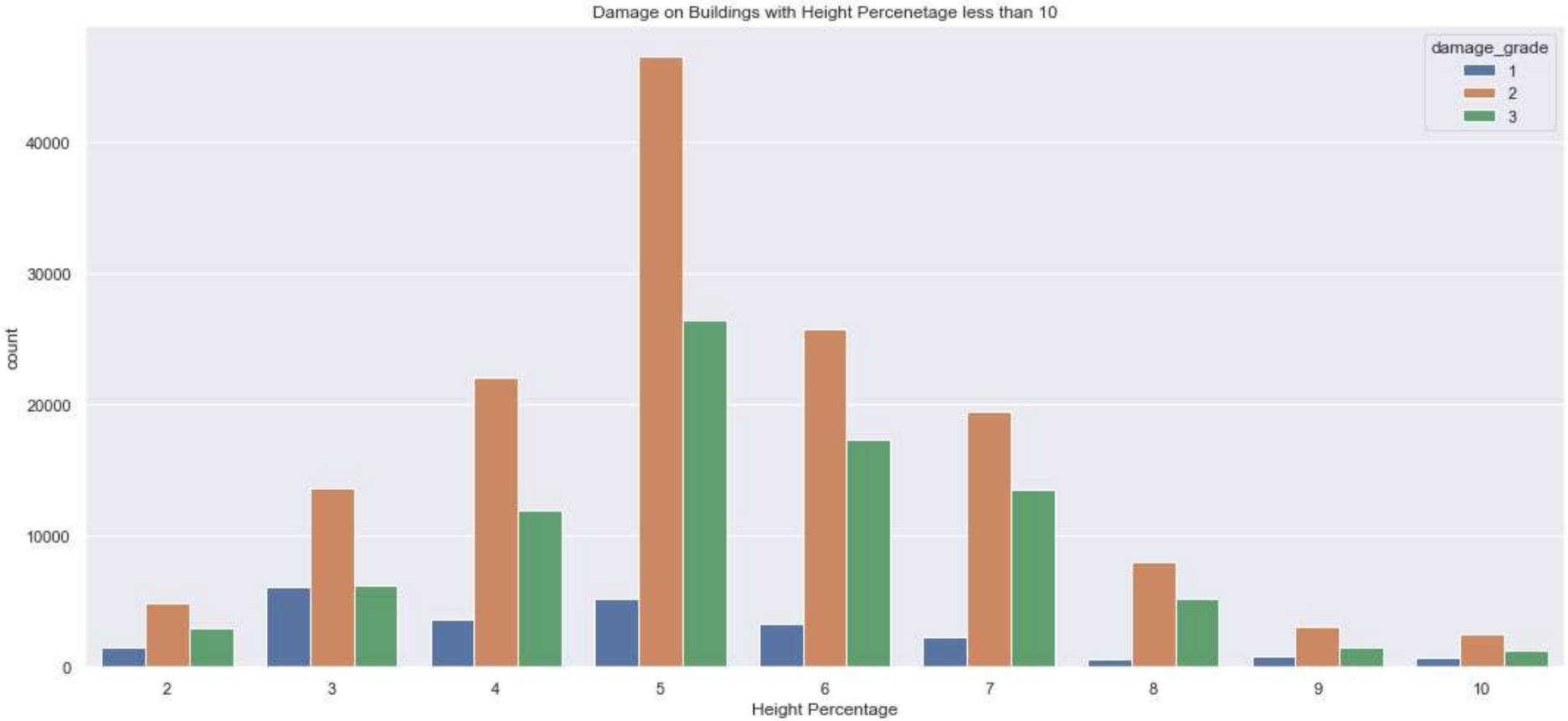
In [37]:

```
plt.figure(figsize=(18,8))
plt.title('Damage on Buildings with Height percentage')
sns.countplot(x='height_percentage',data=data,hue='damage_grade')
plt.xlabel('Height Percentage')
plt.show()
```



In [38]:

```
plt.figure(figsize=(18,8))
plt.title('Damage on Buildings with Height Percenetage less than 10')
buildings_height_percentage_less_than_10 =data[data['height_percentage']<=10]
sns.countplot(x='height_percentage',data=buildings_height_percentage_less_than_10,hue='damage_grade')
plt.xlabel('Height Percentage')
plt.show()
```



- Analysing the damage on building with a particular Height percentage, no special trend was observed.
- For both the Height Percenatage and the Area percentage, the level 2 damage is highest for each value.

In [39]:
```
print("The shape of Dataset after preprocessing is {}".format(data.shape))
```

```
The shape of Dataset after preprocessing is (259211, 42)
```

- After preprocessing, the dataset consists of 259211 rows and 42 columns

In [40]:
```
data.describe(include='all')
```

Out[40]:

	geo_level_1_id	geo_level_2_id	geo_level_3_id	count_floors_pre_eq	age	area_percentage	height_percentage	land_surface_condition	fou
count	259211.000000	259211.000000	259211.000000	259211.000000	259211.000000	259211.000000	259211.000000	259211	
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	t	
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	215522	
mean	13.895664	700.908376	6259.185841	2.129308	21.341706	8.017032	5.433373	NaN	
std	8.029122	412.879348	3647.148813	0.727307	19.606818	4.393804	1.917928	NaN	
min	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	2.000000	NaN	
25%	7.000000	350.000000	3073.000000	2.000000	10.000000	5.000000	4.000000	NaN	
50%	12.000000	702.000000	6272.000000	2.000000	15.000000	7.000000	5.000000	NaN	
75%	21.000000	1050.000000	9412.000000	2.000000	30.000000	9.000000	6.000000	NaN	
max	30.000000	1427.000000	12567.000000	9.000000	200.000000	100.000000	32.000000	NaN	

1. What is the impact of superstructure materials used on the damage level and its correlation with non-engineered and engineered reinforced concrete?

In [41]:

```
# Creating a new dataframe of superstructure material only

buildings_with_different_superstructure_material = data[['has_superstructure_adobe_mud','has_superstructure_mud_mortar_stone',
        'has_superstructure_cement_mortar_stone','has_superstructure_mud_mortar_brick',
        'has_superstructure_cement_mortar_brick', 'has_superstructure_timber',
        'has_superstructure_bamboo', 'has_superstructure_rc_non_engineered',
        'has_superstructure_rc_engineered', 'has_superstructure_other']]
```

In [42]:

```
# Creating a dictionary to store the value count of each the superstructure material used

dictionary_to_store_total_values_of_damage_based_on_material_used = {}
for columns in buildings_with_different_superstructure_material.columns.tolist():
    val = data[columns].value_counts()[1]
    dictionary_to_store_total_values_of_damage_based_on_material_used[columns] = val
```

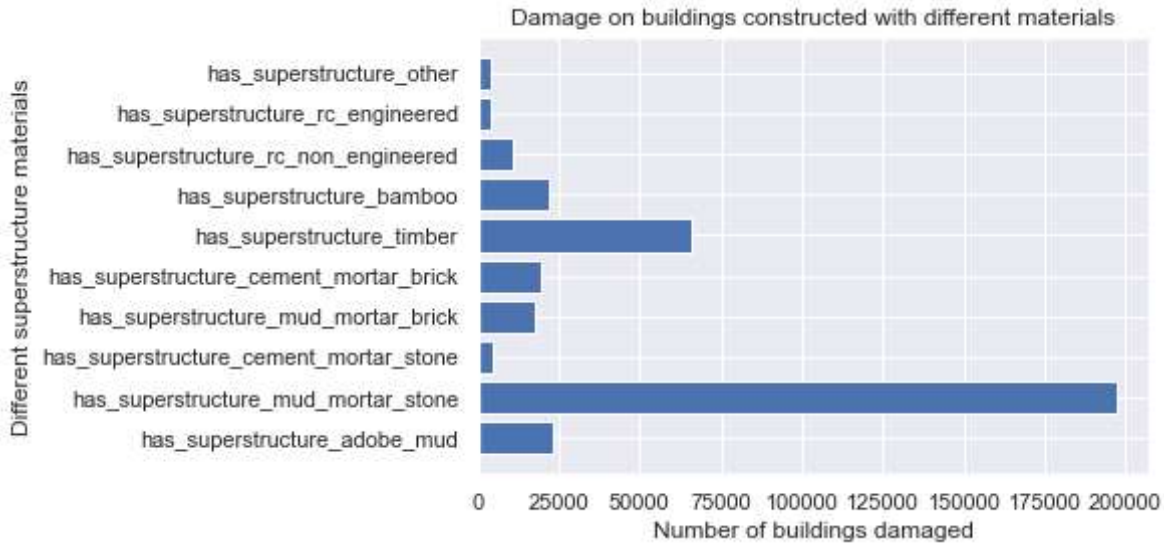
In [43]:
```
# Creating the dataframe from the dictionary created above
total_buildings_damage_on_different_superstructure_material = pd.DataFrame.from_dict(
                                                        dictionary_to_store_total_values_of_damage_based_on_material_used,
                                                        orient='index',columns=['total_damage_on_building'])
```

In [44]:
```
total_buildings_damage_on_different_superstructure_material.total_damage_on_building.sort_values()
```

Out[44]:
```
has_superstructure_other                    3883
has_superstructure_rc_engineered            4113
has_superstructure_cement_mortar_stone      4730
has_superstructure_rc_non_engineered       11054
has_superstructure_mud_mortar_brick        17629
has_superstructure_cement_mortar_brick     19534
has_superstructure_bamboo                  22010
has_superstructure_adobe_mud               22907
has_superstructure_timber                  65969
has_superstructure_mud_mortar_stone       197524
Name: total_damage_on_building, dtype: int64
```

- Displaying the different superstructure material used for number of buildings.

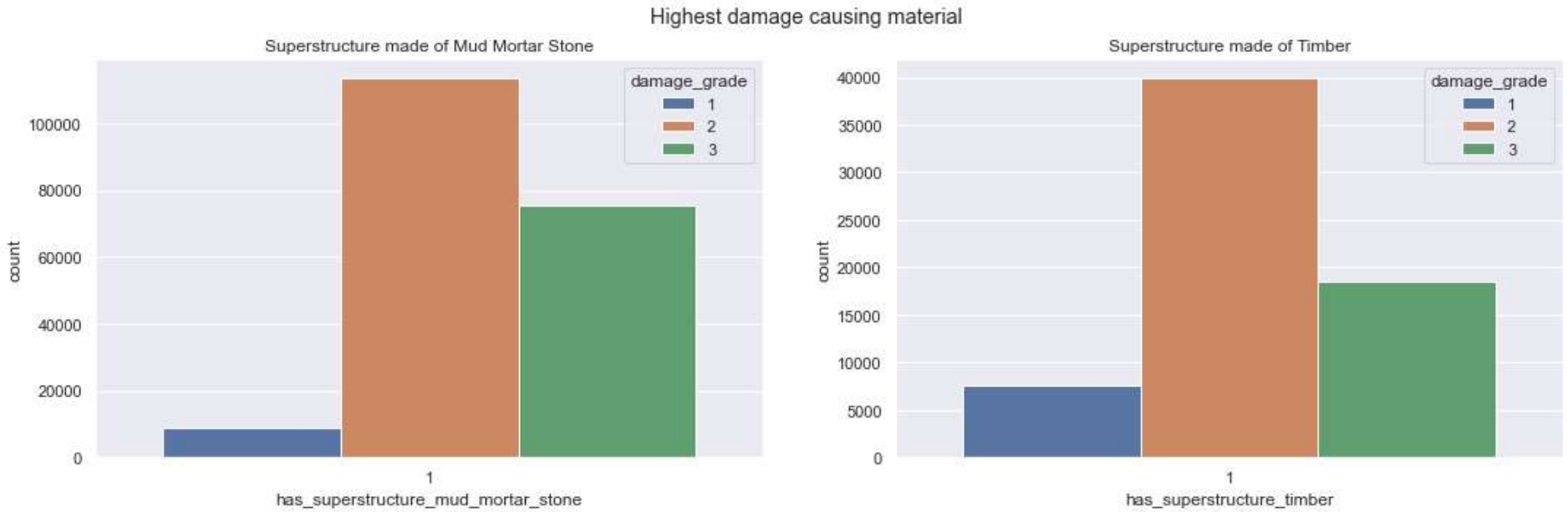
In [45]:
```
# To plot the bar chart for the above dataframe

plt.barh(list(dictionary_to_store_total_values_of_damage_based_on_material_used.keys()),
         list(dictionary_to_store_total_values_of_damage_based_on_material_used.values()))
plt.xlabel('Number of buildings damaged')
plt.ylabel('Different superstructure materials')
plt.title('Damage on buildings constructed with different materials')
plt.show()
```



- In buildings built with materials like Mud mortar stone and Timber has undergone most damage. Therefore, we will plot the countplot of the most damaged superstructure with damage grade to identify any trend or special case.

In [46]:
```
# Creating a dataframes of the superstructure built using the matreials mud mortar stone and timber

buildings_with_mud_mortar_stone = data[data['has_superstructure_mud_mortar_stone']==1]
buildings_with_timber = data[data['has_superstructure_timber']==1]
```

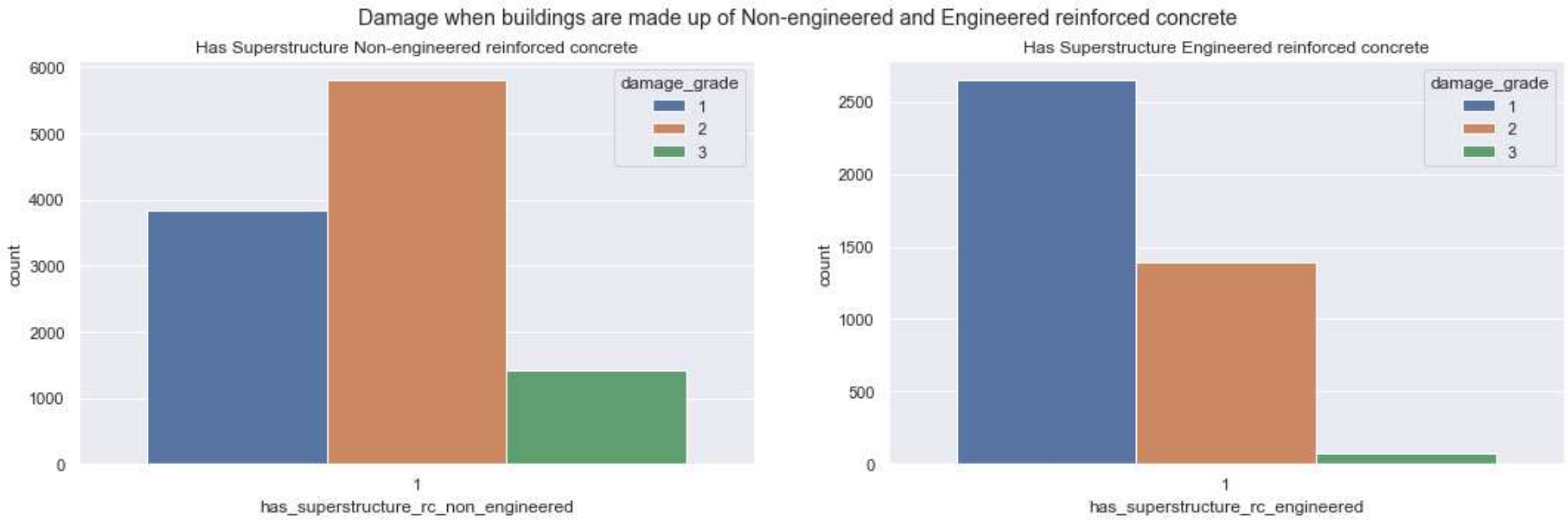
In [47]:
```
fig, axes = plt.subplots(1,2, figsize=(18,5))
fig.suptitle('Highest damage causing material')
sns.countplot(x='has_superstructure_mud_mortar_stone',data=buildings_with_mud_mortar_stone,
              hue='damage_grade',ax=axes[0])
axes[0].set_title('Superstructure made of Mud Mortar Stone')

sns.countplot(x='has_superstructure_timber',data=buildings_with_timber,
              hue='damage_grade',ax=axes[1])
axes[1].set_title('Superstructure made of Timber')

plt.show()
```



- The countplot of superstructure built with both mud motor stone and timber showed similar trend observed before i.e. the maximum number of buildings with damage grade 2

In [48]:

```
fig, axes = plt.subplots(1,2, figsize=(18,5))
fig.suptitle('Damage when buildings are made up of Non-engineered and Engineered reinforced concrete')
sns.countplot(x='has_superstructure_rc_non_engineered',data=data[data['has_superstructure_rc_non_engineered']==1],
              hue='damage_grade',ax=axes[0])
axes[0].set_title('Has Superstructure Non-engineered reinforced concrete')

sns.countplot(x='has_superstructure_rc_engineered',data=data[data['has_superstructure_rc_engineered']==1],
              hue='damage_grade',ax=axes[1])
axes[1].set_title('Has Superstructure Engineered reinforced concrete')

plt.show()
```



- The countplot of damage in buildings made up of Non-engineered and Engineered reinforced concrete indicates that the buildings with engineered reinforced concrete are less likely to cause grade 3 damage and more likely to cause grade 1 damage.

2. How does the plan configuration correlate with the superstructure material being used on the damage level?

In [49]:
```
data.plan_configuration.value_counts()
```

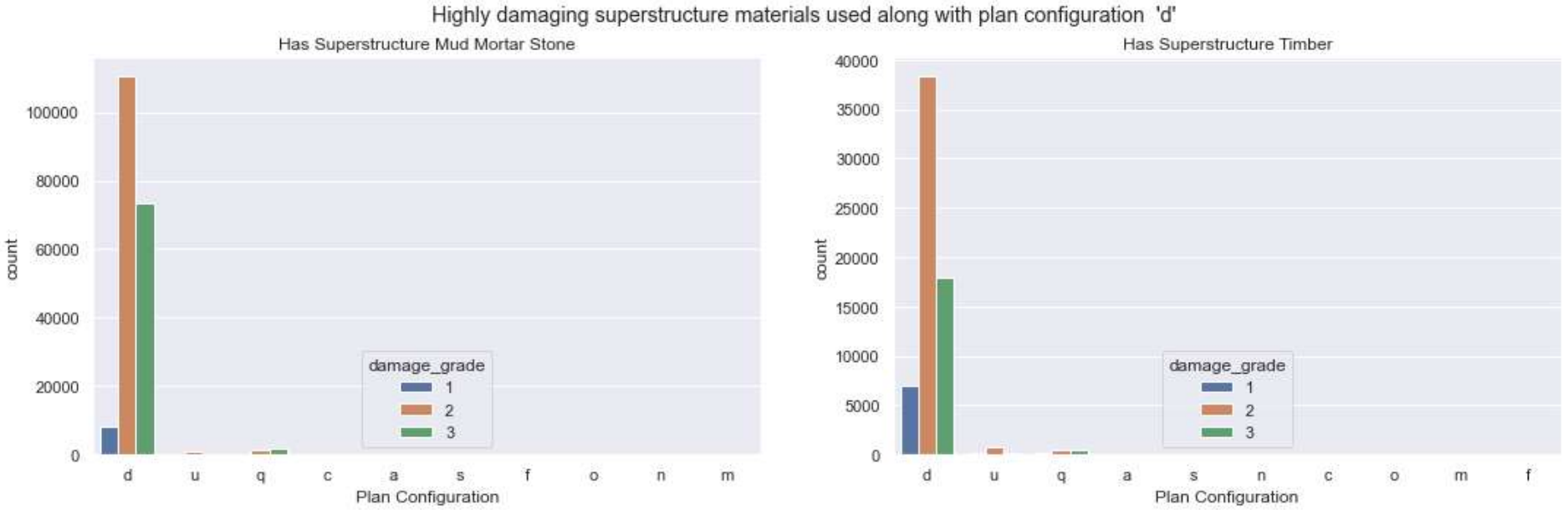
Out[49]:
```
d    248746
q      5656
u      3632
s       344
c       323
a       247
o       159
m        44
n        38
f        22
Name: plan_configuration, dtype: int64
```

In [50]:
```
(data.plan_configuration.value_counts()['d'] * 100)/len(data)
```

Out[50]: 95.96274849446975

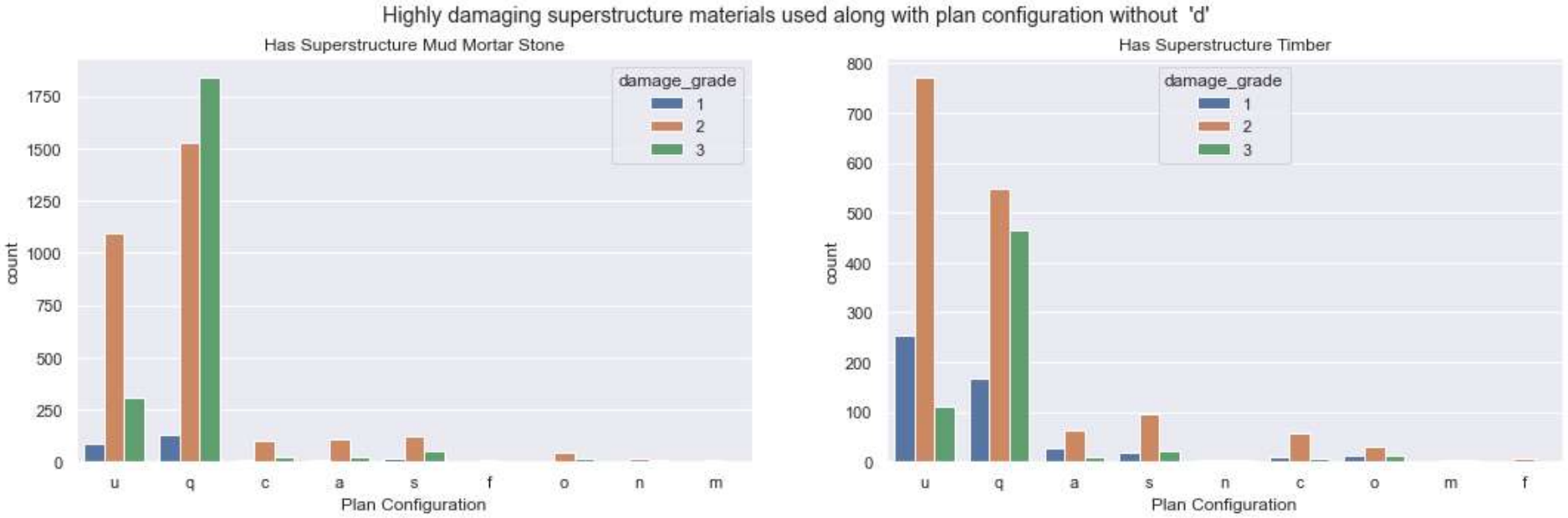
- It is evident that 95.96% of the total buildings have used plan configuration 'd'. Therefore, we will plot several graphs with and without plan configuration 'd' along with the most damaging superstructure materials.

In [51]:
```
fig, axes = plt.subplots(1,2, figsize=(18,5))
fig.suptitle('Highly damaging superstructure materials used along with plan configuration  \'d\'')
sns.countplot(x='plan_configuration',data=buildings_with_mud_mortar_stone, hue='damage_grade',ax=axes[0])
axes[0].set_title('Has Superstructure Mud Mortar Stone')
axes[0].set_xlabel('Plan Configuration')

sns.countplot(x='plan_configuration',data=buildings_with_timber, hue='damage_grade',ax=axes[1])
axes[1].set_title('Has Superstructure Timber')
axes[1].set_xlabel('Plan Configuration')

plt.show()
```



- The countplot of plan configuration d with respect to the most damaging superstructure material, the number of buildings with damage grade 2 was found to be the highest.

In [52]:
```
fig, axes = plt.subplots(1,2, figsize=(18,5))
fig.suptitle('Highly damaging superstructure materials used along with plan configuration without  \'d\'')
sns.countplot(x='plan_configuration',data=buildings_with_mud_mortar_stone[buildings_with_mud_mortar_stone['plan_configuration'] !=
              hue='damage_grade',ax=axes[0])
axes[0].set_title('Has Superstructure Mud Mortar Stone')
axes[0].set_xlabel('Plan Configuration')

sns.countplot(x='plan_configuration',data=buildings_with_timber[buildings_with_timber['plan_configuration'] !='d'],
              hue='damage_grade',ax=axes[1])
axes[1].set_title('Has Superstructure Timber')
axes[1].set_xlabel('Plan Configuration')

plt.show()
```



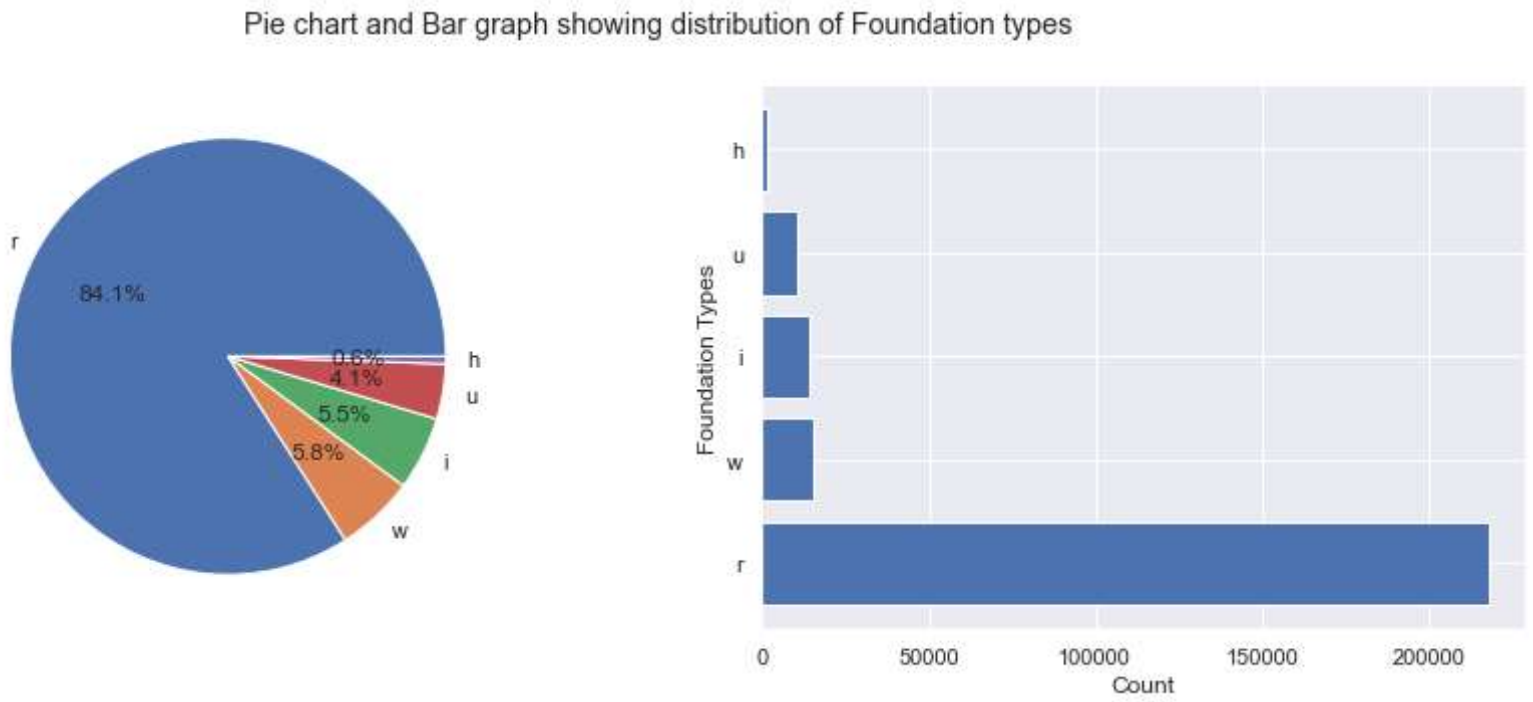
- By plotting the countplot of plan configuration without 'd' along with the highly damaging superstructure material, it is observed that more number of buildings are damaged with grade 3 when built with plan configuration 'q' and superstructure Mud Mortar Stone. Conversely, when plan configuration 'q' is used with superstructure Timber, more number of buildings are damaged with grade 2, however, there is no significant difference in the number of buildings with damage grade 2 and 3.

3. Which type of foundation is more prone to damage?

In [53]:
```
data.foundation_type.value_counts()
```

Out[53]:
```
r    217932
w     15108
u     14165
i     10558
h      1448
Name: foundation_type, dtype: int64
```

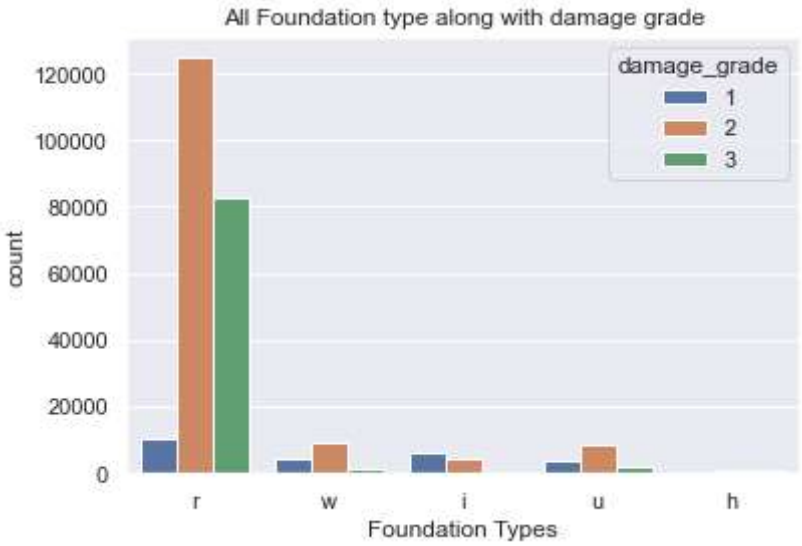
In [54]:
```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))
fig.suptitle('Pie chart and Bar graph showing distribution of Foundation types')
axs[0].pie(data.foundation_type.value_counts(),labels=data.foundation_type.unique(), autopct='%1.1f%%')
axs[1].barh(data.foundation_type.unique(), data.foundation_type.value_counts())
axs[1].set_xlabel('Count')
axs[1].set_ylabel('Foundation Types')
plt.show()
```



- Intotal, 84.1% of the buildings have used Foundation type 'r'.

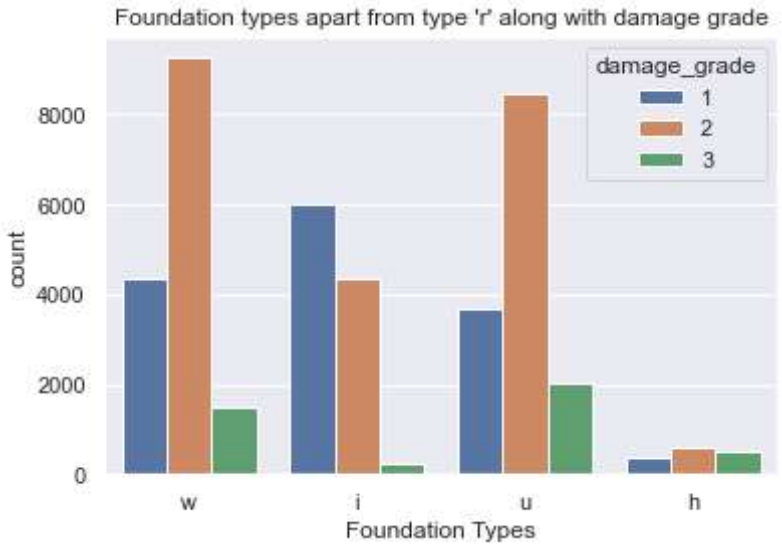
- Therefore, we will plot several graphs with and without Foundation type 'r' along with respect to damage grade to identify most damage causing foundation type.

In [55]:
```
sns.countplot(x='foundation_type',hue='damage_grade',data=data)
plt.title('All Foundation type along with damage grade')
plt.xlabel('Foundation Types')
plt.show()
```



- By plotting the countplot of different foundation types, it is observed that the building with foundation type 'r' is most likely to cause level 2 damage.

In [56]:
```
foundation_type_without_r = data[data['foundation_type'] !='r']
sns.countplot(x='foundation_type',hue='damage_grade',data=foundation_type_without_r)
plt.title('Foundation types apart from type \'r\' along with damage grade')
plt.xlabel('Foundation Types')
plt.show()
```



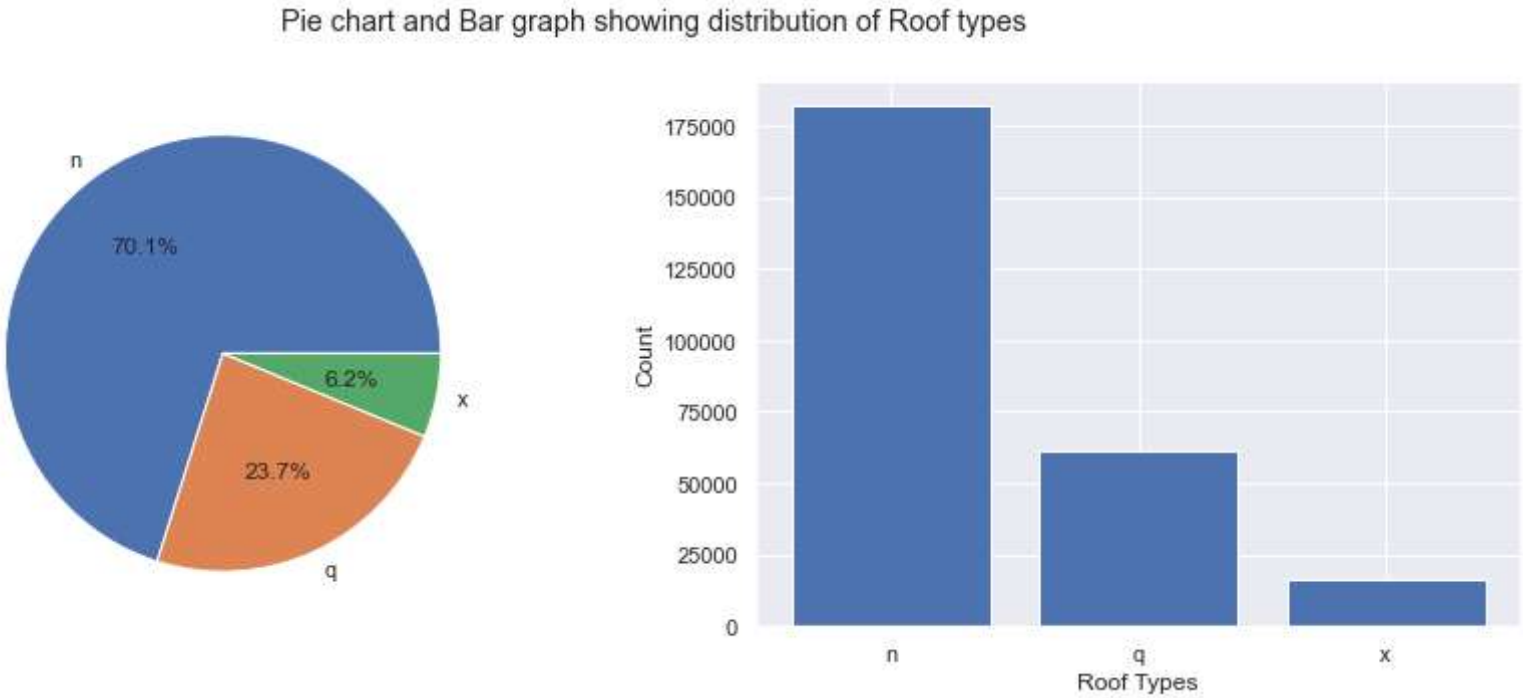
- By plotting the countplot of foundation types without 'r' we observed that foundation type 'i' is most likely to cause level 1 damage. Therefore, foundation type 'i' could be considerd one of the safest.

4. Which roof type will cause a high level of damage?

In [57]:
```
data.roof_type.value_counts()
```

Out[57]:
```
n    181762
q     61333
x     16116
Name: roof_type, dtype: int64
```

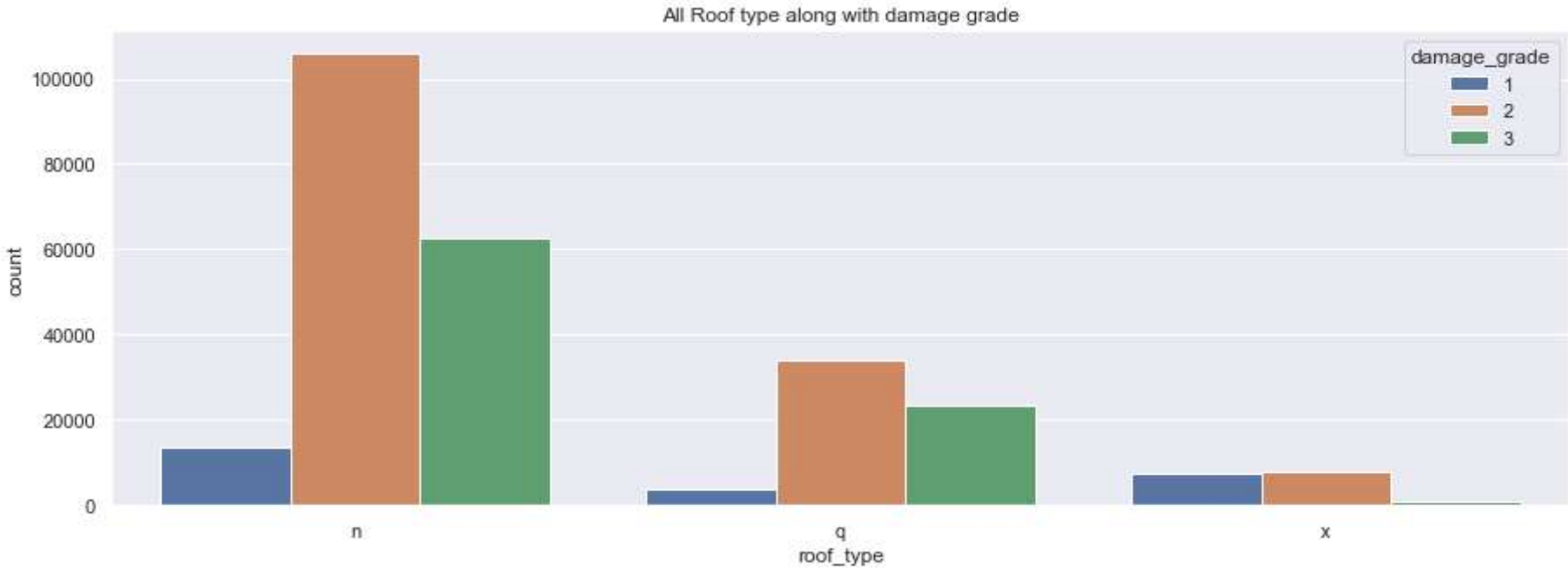
In [58]:
```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))
fig.suptitle('Pie chart and Bar graph showing distribution of Roof types')
axs[0].pie(data.roof_type.value_counts(),labels=data.roof_type.unique(), autopct='%1.1f%%')
axs[1].bar(data.roof_type.unique(), data.roof_type.value_counts())
axs[1].set_xlabel('Roof Types')
axs[1].set_ylabel('Count')
plt.show()
```



- It is evident that 70.1% of the total buildings have used roof type 'n'.

In [59]:

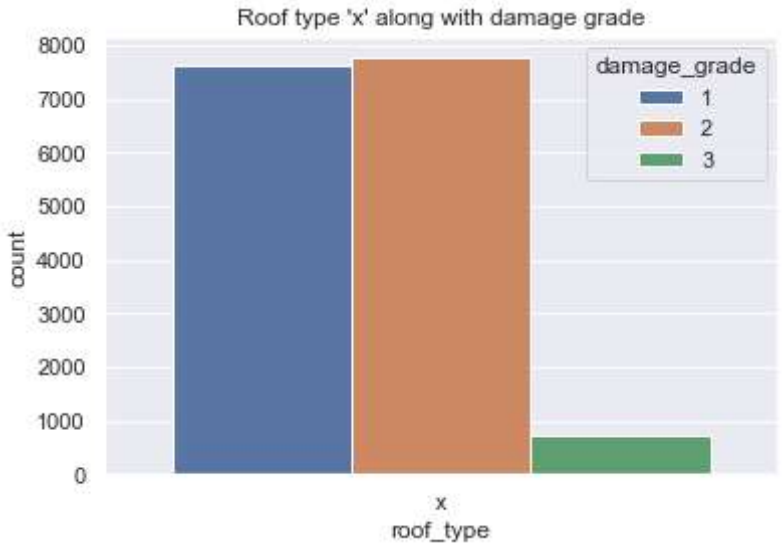
```
plt.figure(figsize=(15, 5))
sns.countplot(x='roof_type',hue='damage_grade',data=data)
plt.title('All Roof type along with damage grade')
plt.show()
```



- By plotting the countplot of different roof types along with damage grade we can say that roof types 'n' and 'q' follows general trend of the dataset in which the level 2 damage is higher than any other level of damage. For roof type 'x' we will plot a diffrent countplot as there is not much difference between level 1 and level 2 damage.

In [60]:

```
sns.countplot(x='roof_type',hue='damage_grade',data=data[data.roof_type=='x'])
plt.title('Roof type \'x\' along with damage grade')
plt.show()
```



- When roof type 'x' is used, the building are least expected to cause level 3 damage and there is hardly any diffrence in the number of buildings with level 1 and level 2 damage.

5. Which combination of roof type and foundation type will be suitable for the development of most earthquake-resilient buildings?

To answer above question we will use dataframe.groupby() and dataframe.unstack() method. so that we have combination of foundation type and roof type to answer our question.

In [61]:
```
dataframe_grouped_by_roof_and_foundation =  data.groupby(['roof_type','foundation_type'])
dataframe_grouped_by_roof_and_foundation['damage_grade'].value_counts().unstack()
```

Out[61]:

damage_grade		1	2	3
roof_type	foundation_type			
n	h	189	454	369
	i	335	508	32
	r	8456	95592	60098
	u	1712	4714	1255
	w	2738	4496	814

roof_type	foundation_type	damage_grade 1	2	3
q	h	155	106	140
	i	32	83	10
	r	1631	27496	22173
	u	470	1447	530
	w	1608	4772	680
x	h	14	19	2
	i	5624	3755	179
	r	501	1676	309
	u	1480	2319	238

- From the above graphs we can say that there is no building in the dataset with combination of roof type 'x' and foundation type'w'. As per the observation, we can assume that this combination is safe or there exist no building with this combination.
- From the given datset we can say that roof type 'x' and foundation type 'i' is safe because the numbers of buildings with damage grade level 1 is the highest (5624) followed by damage grade level 2 (3755) and damage grade level 3 (179).

6. What is the height percentage and count of floors impact on damage level?

In [62]:
```
data.count_floors_pre_eq.value_counts()
```

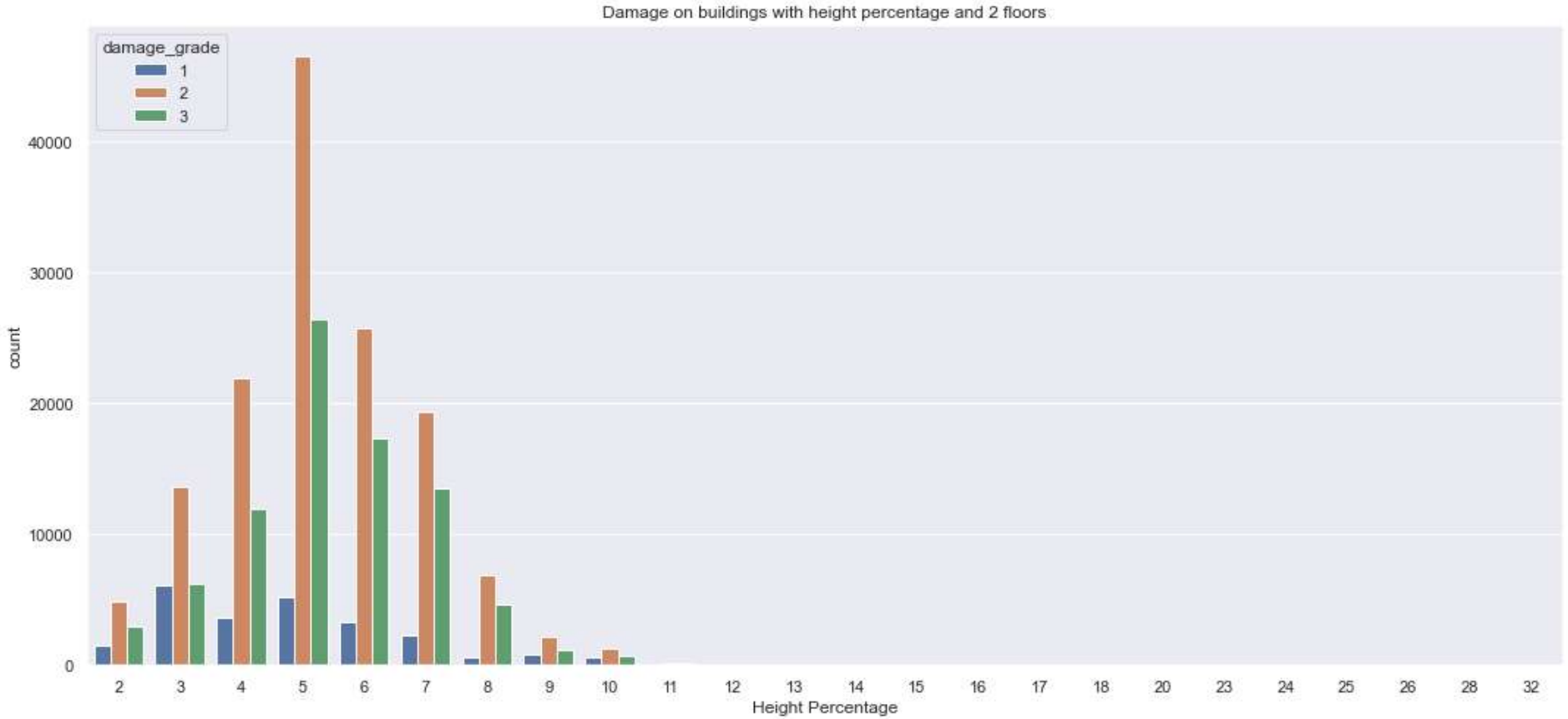
Out[62]:
```
2    155754
3     55332
1     40272
4      5390
5      2218
6       204
7        39
8         1
9         1
Name: count_floors_pre_eq, dtype: int64
```

In [63]:
```
# Creating a dataframe with atmost 3 floors in the building.

buildings_with_atmost_3_floors = data[data['count_floors_pre_eq']<=3]
(len(buildings_with_atmost_3_floors) * 100) / len(data)
```

Out[63]: 96.9704217799399

- 97% of buildings has less than or equal to 3 floors. Hence, we will be using buildings with less than 4 floors further to see whether we can identify any trend.

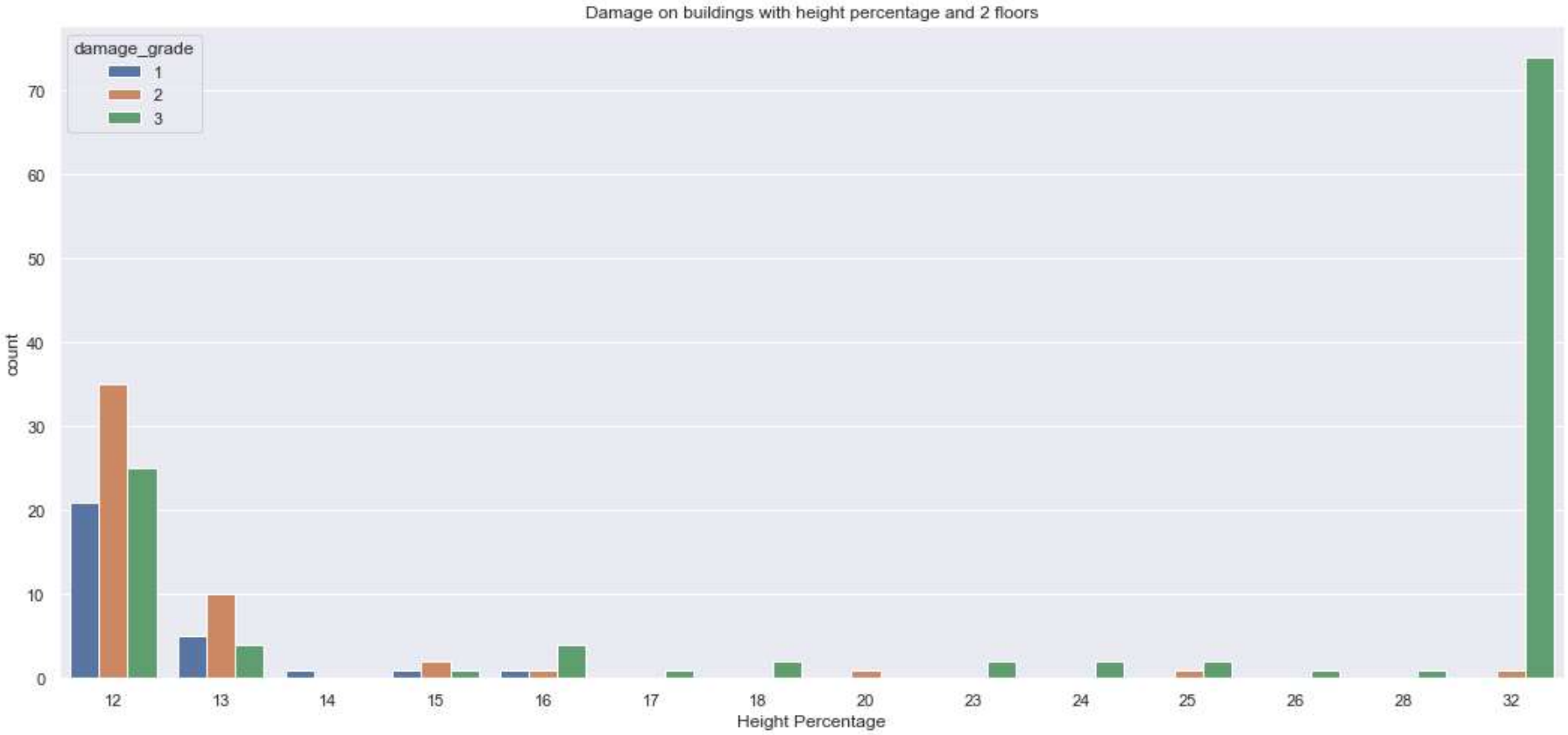
In [64]:
```
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with height percentage and 2 floors')
sns.countplot(x='height_percentage',data=buildings_with_atmost_3_floors,hue='damage_grade')
plt.xlabel('Height Percentage')
plt.show()
```



- In the above countplot, the again the repetitive trend observed the dataset is evident,i.e. the damage grade 2 is higher than other damage grades irrespective of the floors. We will plot another countplot where height percentage is higher than 11.

In [65]:

```
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with height percentage and 2 floors')
sns.countplot(x='height_percentage',data=buildings_with_atmost_3_floors[buildings_with_atmost_3_floors.height_percentage > 11],
              hue='damage_grade')
plt.xlabel('Height Percentage')
plt.show()
```

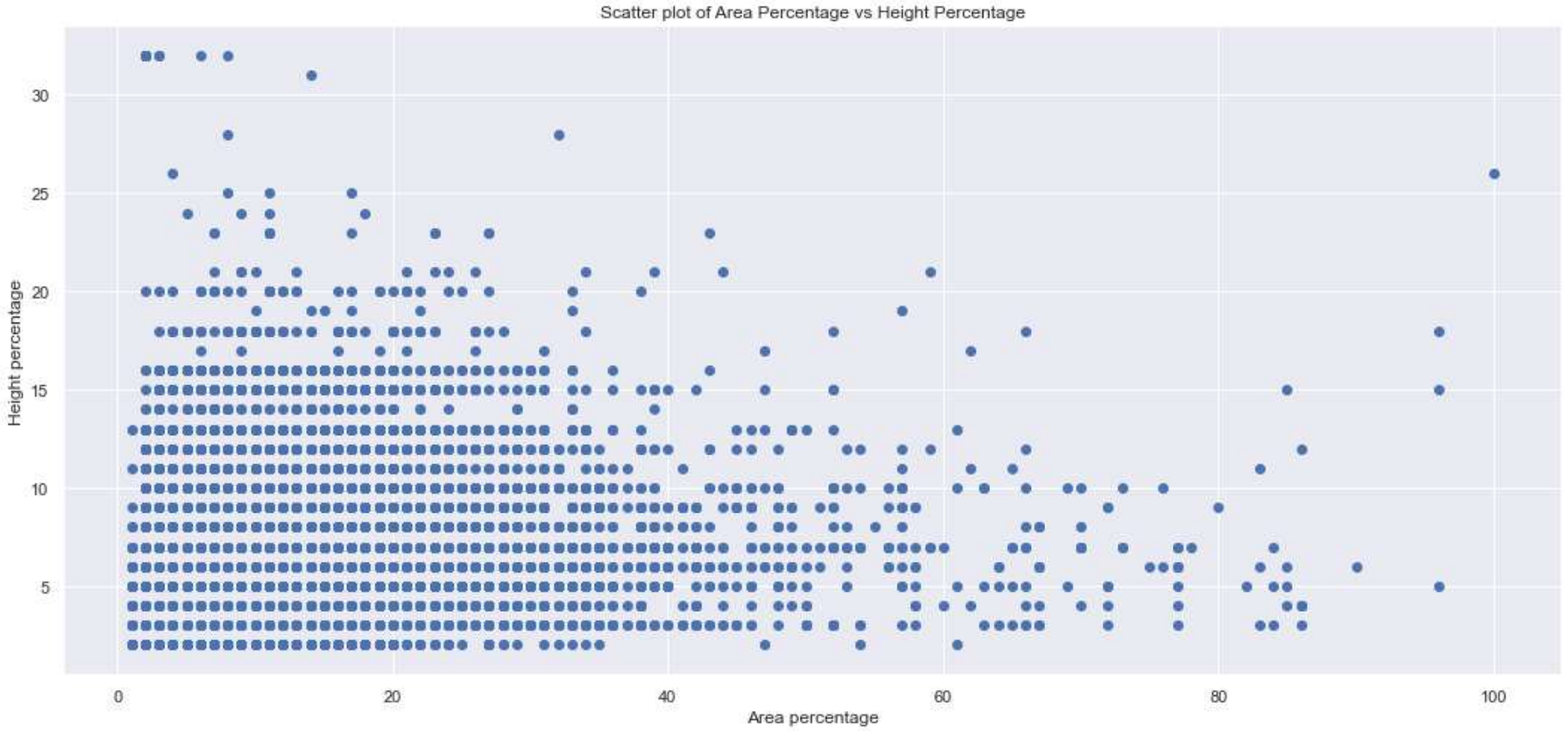


- Height percentage 16 onwards, the level 3 damage is more than any other. However, we cannot confirm this is trend since the number of buidings with height percent greater than 15 are comparatively very less.

7. Does the area percentage to height percentage ratio affect the damage level?

In [66]:

```
plt.figure(figsize=(18,8))
plt.scatter(x = data.area_percentage, y = data.height_percentage)
plt.xlabel('Area percentage')
plt.ylabel('Height percentage')
plt.title('Scatter plot of Area Percentage vs Height Percentage')
plt.show()
```



- Scatter plot of area vs height percentage shows that the more buildings whose area to height ratio would be less then 2 as we have biggest cluster around in from (0,0) to (40,20).

In [67]:
```
# Creating a new dataframe using dataframe.copy()

data_copy = data.copy()

# Adding a new feature area ratio height

data_copy['area_ratio_height'] = data_copy.area_percentage / data_copy.height_percentage
```

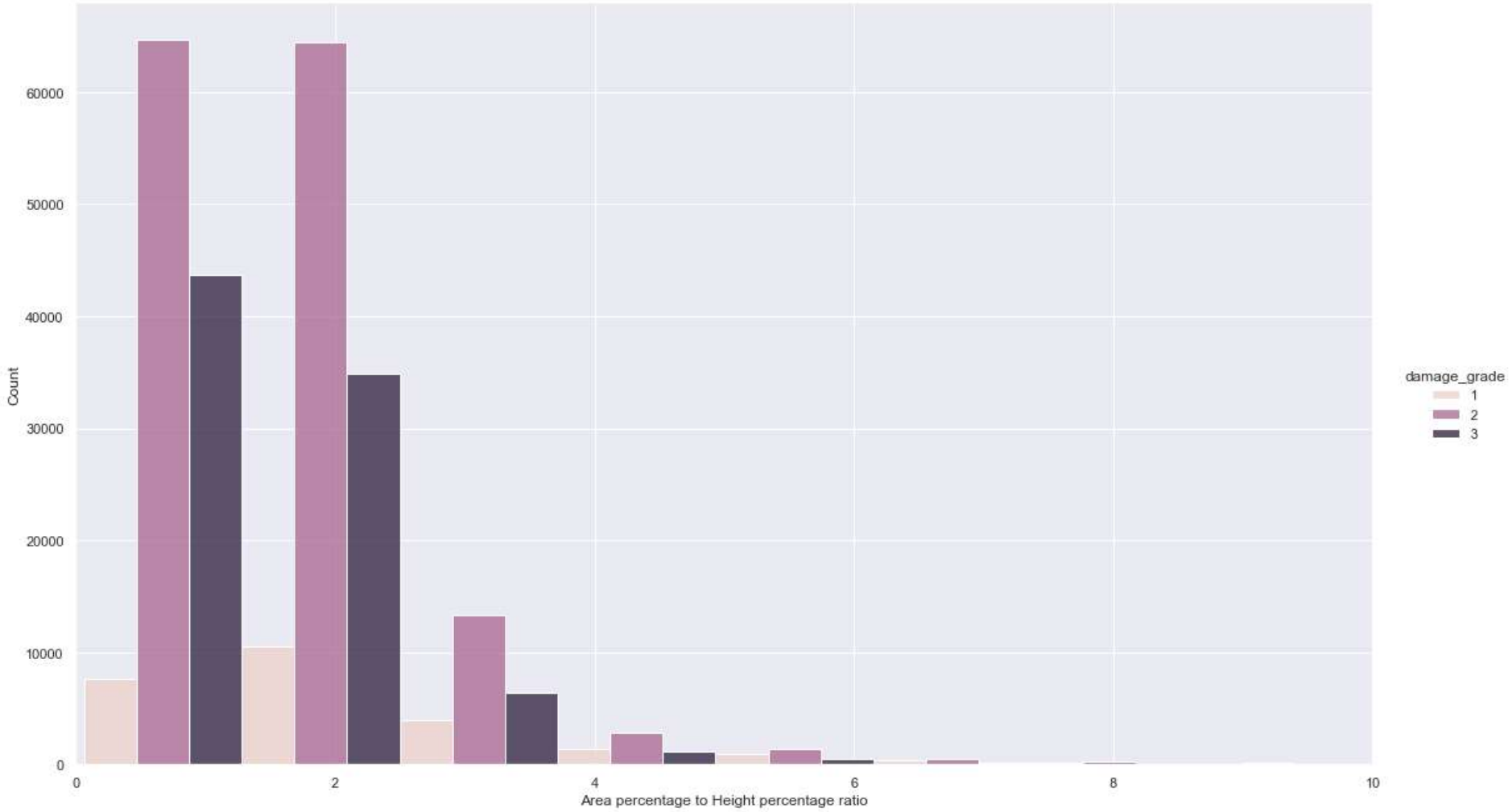
In [68]:
```
data_copy['area_ratio_height'].nunique()
```

Out[68]: 511

- There are total 511 unique values in the new feature (area_ratio_height).

In [69]:
```
plt.figure(figsize=(18,8))
sns.displot(data=data_copy, x="area_ratio_height", hue="damage_grade",multiple='dodge',bins=25,height=9.27, aspect=15.7/9.27)
plt.xlabel('Area percentage to Height percentage ratio')
plt.xlim(0,10)
plt.show()
```

```
<Figure size 1296x576 with 0 Axes>
```

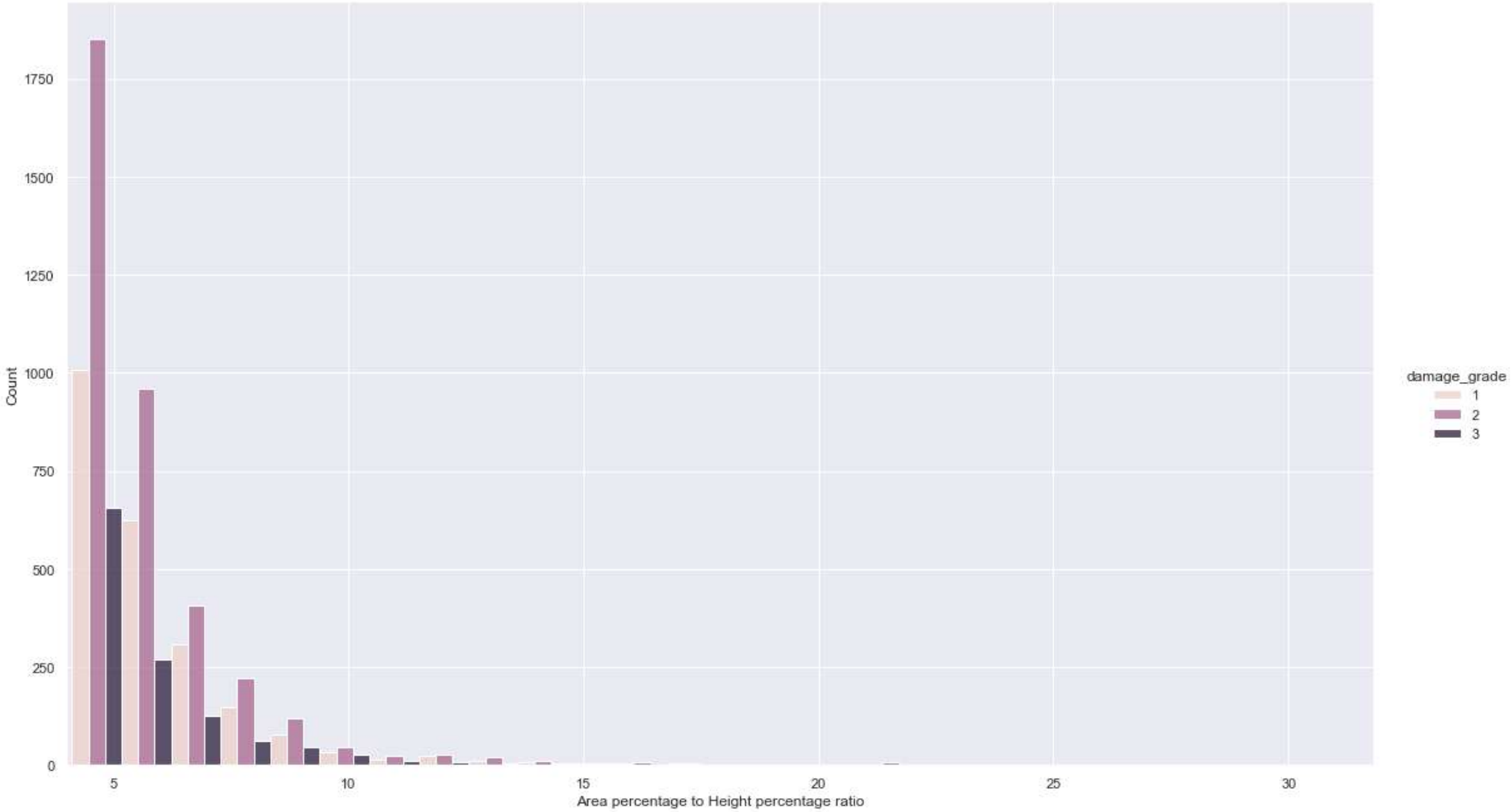


- As we can see from the countplot, mostly general trend followed, which is damage grade 2 higher than other grades irrespective of the area to height ratio. We will plot another countplot where area to height ratio is higher than 4.

In [70]:

```
plt.figure(figsize=(18,8))
buildings_with_area_ratio_height_greater_than_4 = data_copy[data_copy['area_ratio_height']>4]
sns.displot(data=buildings_with_area_ratio_height_greater_than_4, x="area_ratio_height", hue="damage_grade",
            multiple='dodge',bins=25,height=9.27, aspect=15.7/9.27)
plt.xlim(4)
plt.xlabel('Area percentage to Height percentage ratio')
plt.show()
```

```
<Figure size 1296x576 with 0 Axes>
```



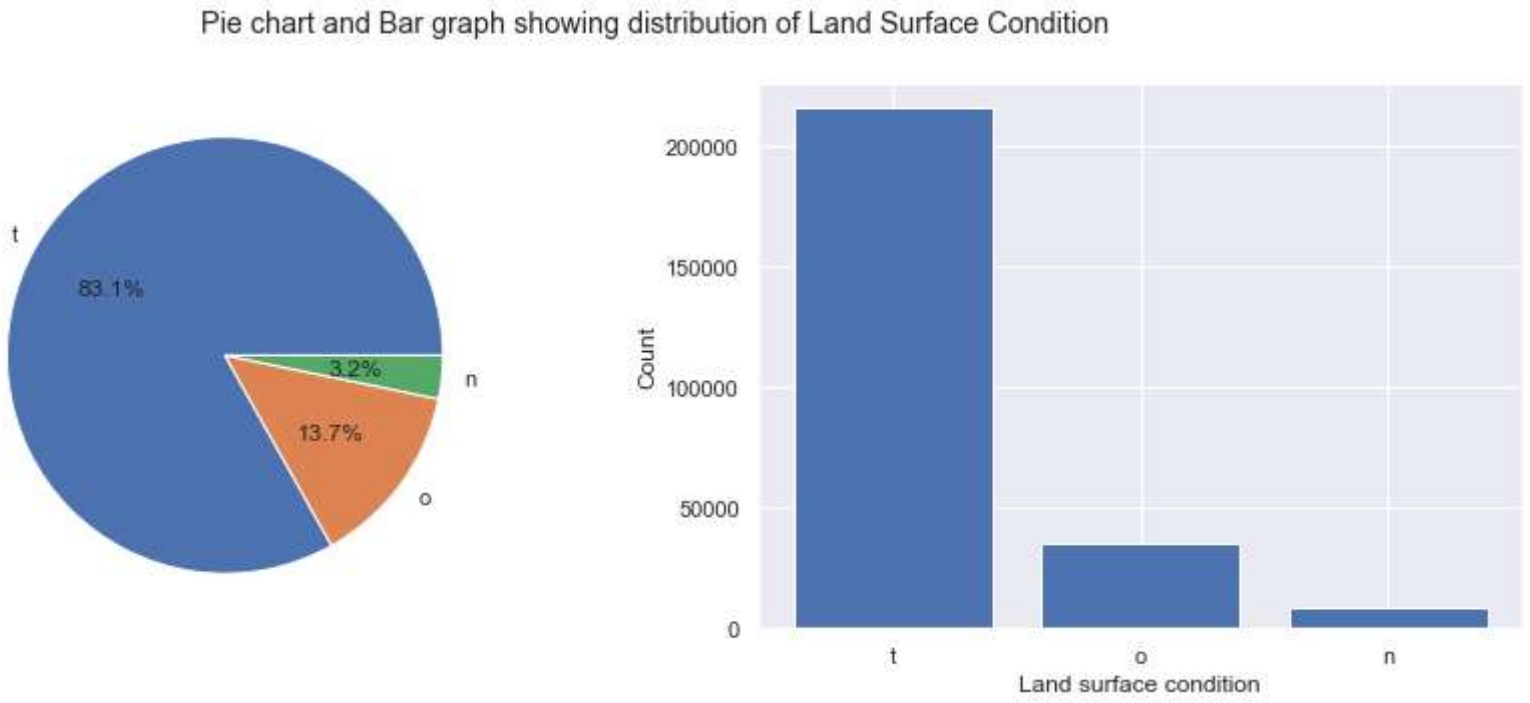
- With the area to height percent ratio greater than 4, level 1 damage is more than level 3, we can't say this is trend because the there are less buildings with area to height percent ratio more than 4. However, with the domain knowledge one can say this is quite normal wider building with less height is less likely to get damaged.

8. How does the combination of land surface condition and the age of the building affect the damage?

In [71]:
```
data.land_surface_condition.value_counts()
```

Out[71]:
```
t    215522
n     35389
o      8300
Name: land_surface_condition, dtype: int64
```

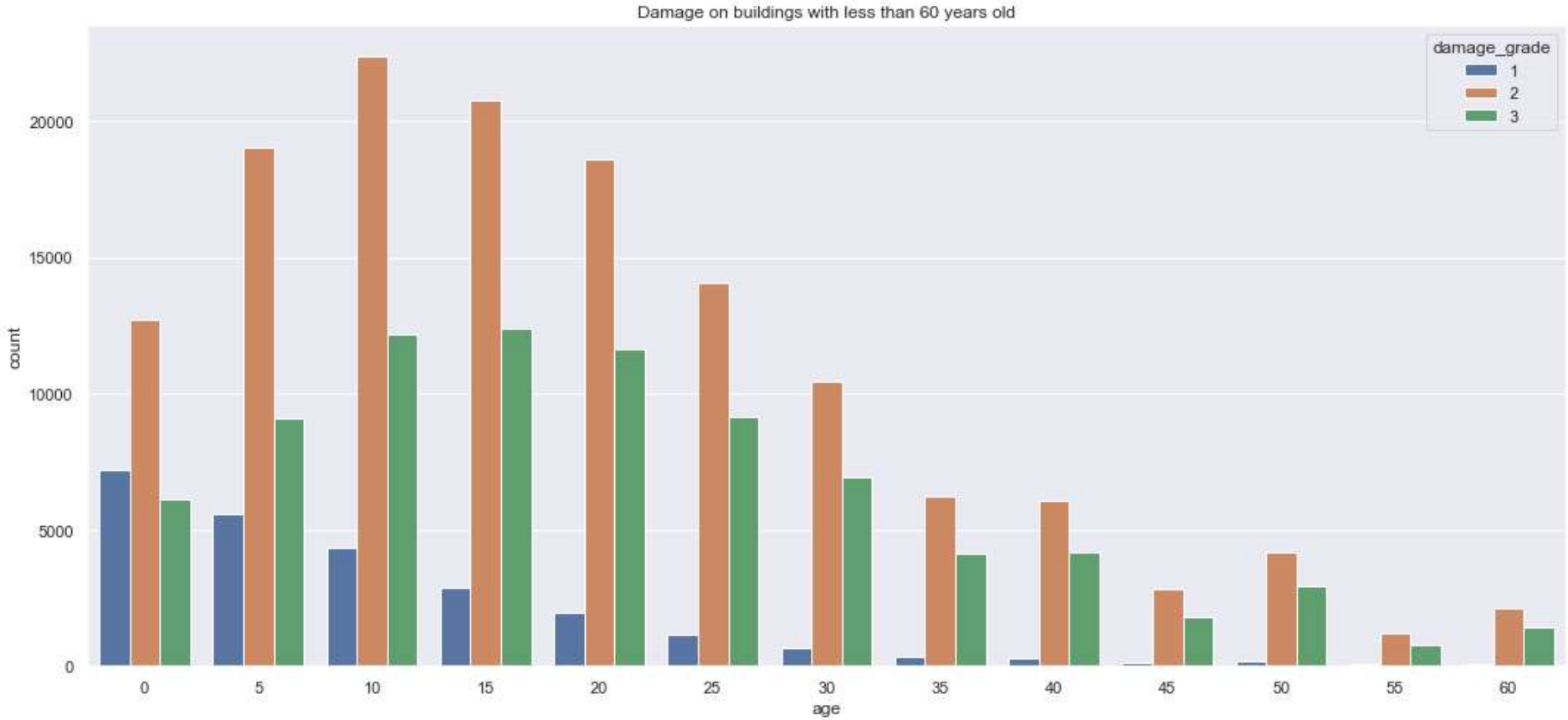
In [72]:
```
fig, axs = plt.subplots(1, 2, figsize=(15, 5))
fig.suptitle('Pie chart and Bar graph showing distribution of Land Surface Condition')
axs[0].pie(data.land_surface_condition.value_counts(),labels=data.land_surface_condition.unique(), autopct='%1.1f%%')
axs[1].bar(data.land_surface_condition.unique(), data.land_surface_condition.value_counts())
axs[1].set_xlabel('Land surface condition')
axs[1].set_ylabel('Count')
plt.show()
```

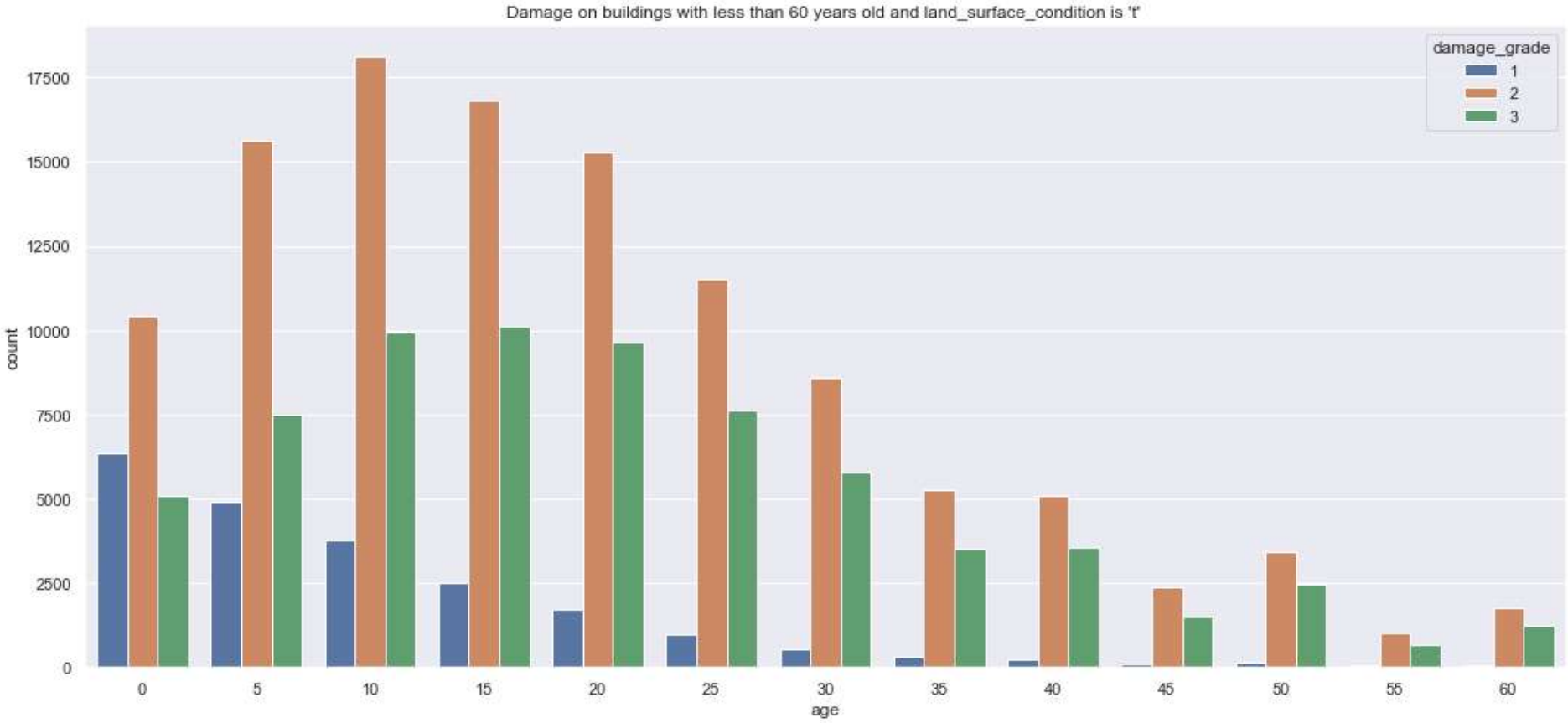


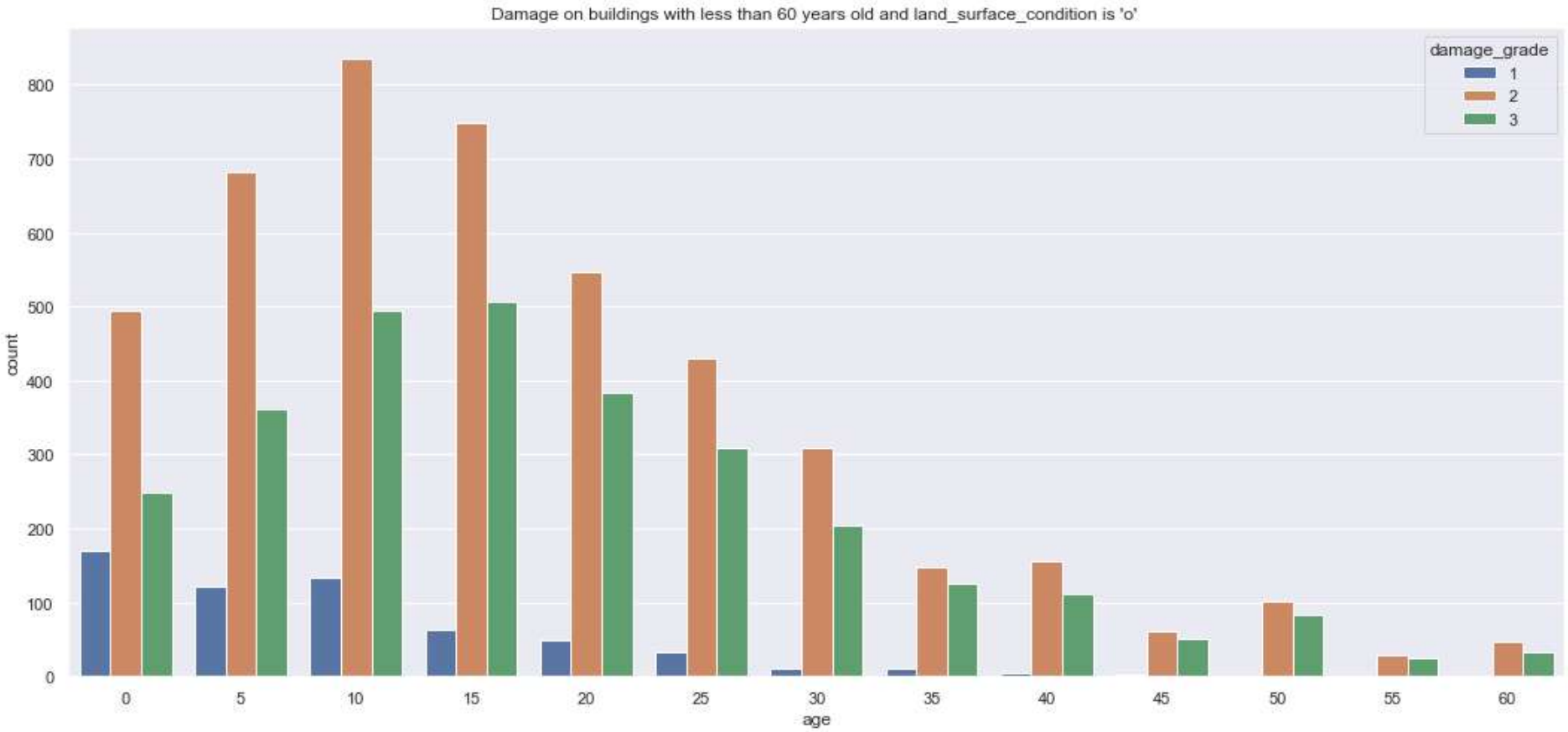
- 83.1% and 13.7% of the total buildings have land surface condition 't' and 'o' respectively.
- Therefore, we will plot several graphs with and without Land surface condition 't' and 'o' with age less than 60 and gain insights.

In [73]:
```
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with less than 60 years old')
```

```
buildings_age_less_than_60 =data[data['age']<=60]
sns.countplot(x='age',data=buildings_age_less_than_60,hue='damage_grade')
plt.show()

data_with_land_surface_condition_t = data[data.land_surface_condition =='t']
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with less than 60 years old and land_surface_condition is \'t\' ')
buildings_age_less_than_60_t =data_with_land_surface_condition_t[data_with_land_surface_condition_t['age']<=60]
sns.countplot(x='age',data=buildings_age_less_than_60_t,hue='damage_grade')
plt.show()

data_with_land_surface_condition_n = data[data.land_surface_condition =='o']
plt.figure(figsize=(18,8))
plt.title('Damage on buildings with less than 60 years old and land_surface_condition is \'o\' ')
buildings_age_less_than_60_n =data_with_land_surface_condition_n[data_with_land_surface_condition_n['age']<=60]
sns.countplot(x='age',data=buildings_age_less_than_60_n,hue='damage_grade')
plt.show()
```


Damage on buildings with less than 60 years old
damage_grade
1
2
3
count
20000
15000
10000
5000
0
0
5
10
15
20
25
30
35
40
45
50
55
60
age


Damage on buildings with less than 60 years old and land_surface_condition is 't'
damage_grade
1
2
3
count
17500
15000
12500
10000
7500
5000
2500
0
0
5
10
15
20
25
30
35
40
45
50
55
60
age



- Unfortunately we couldn't find any trend after plotting 3 different graphs based on conditions above.

9. Is the building used for any secondary purpose? If yes, what type of secondary use of the building is more prone to earthquake damage?

In [74]:
```
# Creating a dataframe of secondary usage only

buildings_with_diffrent_secondary_usages = data_input[['has_secondary_use_agriculture','has_secondary_use_hotel',
        'has_secondary_use_rental','has_secondary_use_institution', 'has_secondary_use_school', 'has_secondary_use_industry',
       'has_secondary_use_health_post', 'has_secondary_use_gov_office',
       'has_secondary_use_use_police', 'has_secondary_use_other']]
```

In [75]:
```
# Creating a dictionary to store the value count of each the type of secondary uasge
```

```
dictionary_to_store_total_values_of_damage_based_on_secondary_usage = {}
for columns in buildings_with_diffrent_secondary_usages.columns.tolist():
    val = data[columns].value_counts()[1]
    dictionary_to_store_total_values_of_damage_based_on_secondary_usage[columns] = val
```

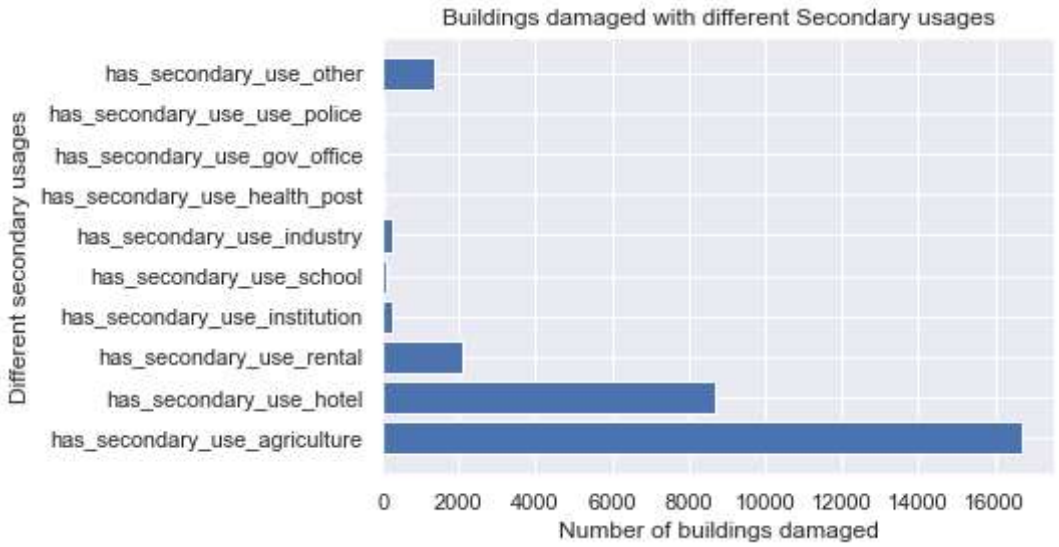
In [76]:
```
# Creating the dataframe from the dictionary created above

total_buildings_damage_on_diffrent_secondary_usages = pd.DataFrame.from_dict(
                                                      dictionary_to_store_total_values_of_damage_based_on_secondary_usage,
                                                      orient='index',columns=['total_damage_on_building'])
```

In [77]:
```
total_buildings_damage_on_diffrent_secondary_usages.total_damage_on_building.sort_values()
```

Out[77]:
```
has_secondary_use_use_police        23
has_secondary_use_gov_office        38
has_secondary_use_health_post       49
has_secondary_use_school            94
has_secondary_use_institution      244
has_secondary_use_industry         279
has_secondary_use_other           1331
has_secondary_use_rental          2087
has_secondary_use_hotel           8710
has_secondary_use_agriculture    16704
Name: total_damage_on_building, dtype: int64
```

- Displaying the number of buildings used for different secondary purposes. Plotting the same dataframe in bar chart to get the clear idea.

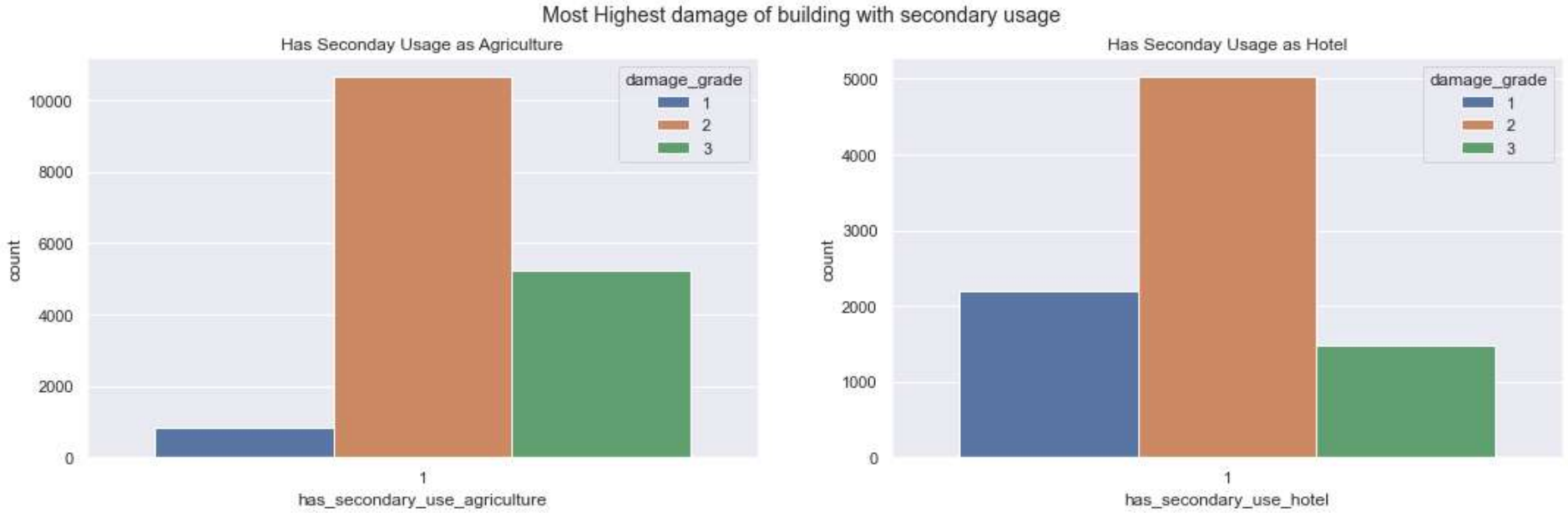
In [78]:
```
plt.barh(list(dictionary_to_store_total_values_of_damage_based_on_secondary_usage.keys()),
         list(dictionary_to_store_total_values_of_damage_based_on_secondary_usage.values()))
plt.xlabel('Number of buildings damaged')
plt.ylabel('Different secondary usages')
plt.title('Buildings damaged with different Secondary usages')
plt.show()
```



- Buildings used secondarily for agriculture and hotel observed the most damage. We will plot the countplot of both the most damaged secondary usage buildings with damage grade to see wether we can find any trend or special case.

In [79]:

```
fig, axes = plt.subplots(1,2, figsize=(18,5))
fig.suptitle('Most Highest damage of building with secondary usage')
sns.countplot(x='has_secondary_use_agriculture',data=data[data['has_secondary_use_agriculture']==1]
              ,hue='damage_grade',ax=axes[0])
axes[0].set_title('Has Seconday Usage as Agriculture')

sns.countplot(x='has_secondary_use_hotel',data=data[data['has_secondary_use_hotel']==1]
              ,hue='damage_grade',ax=axes[1])
axes[1].set_title('Has Seconday Usage as Hotel')


plt.show()
```



- Unfortunately, no particular trend or special case for most damaged secondary usage buildings is observed.

9. Which combination of ground floor type and other floor type will be suitable for the development of most earthquake-resilient buildings?

To answer above question we will use dataframe.groupby() and dataframe.unstack() method. so that we have combination of ground floor type and other floor type to answer our question.

In [80]:
```
dataframe_grouped_by_floor_type =  data.groupby(['ground_floor_type','other_floor_type'])
dataframe_grouped_by_floor_type['damage_grade'].value_counts().unstack()
```

Out[80]:

	damage_grade	1	2	3
ground_floor_type	**other_floor_type**			
f	**j**	3533	13245	8950
	q	6224	86281	53767
	s	84	440	138
	x	2506	19205	14048
m	**j**	45	75	16

	damage_grade	1	2	3
ground_floor_type	other_floor_type			
	q	16	145	37
	s	21	24	5
	x	8	98	17
v	j	4393	4616	280
	q	388	2027	329
	s	4884	4774	443
	x	600	1484	258
x	j	777	2170	1186
	q	634	9107	5121
	s	405	635	112
	x	229	2580	1853
z	j	108	170	117
	q	27	169	39
	s	6	4	2
	x	57	188	111

- From the above table we can say that ground floor type 'v' and other floor type 'j' or 's' is causing less damaged to the buildings in the earthquake as both has less than 10% of level 3 damage as compared to level 1 or level 2 damage. In addition, there is no big difference between level 1 damage and level 2 damage for this combination.