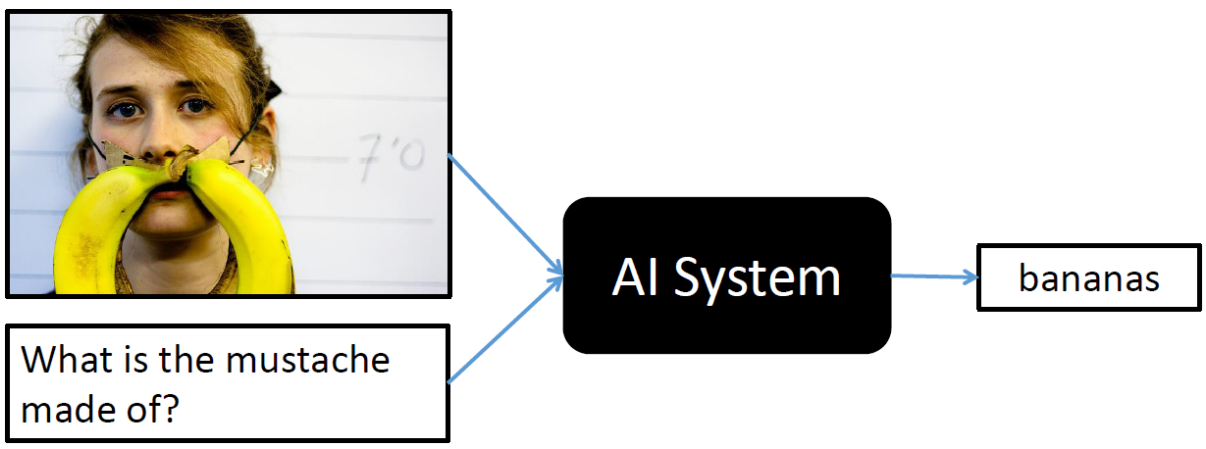
SELF CASE STUDY 2 : VISUAL QUESTION ANSWERING

1.1 PROBLEM STATEMENT

Visual Question Answering(VQA) is a system that takes an image and natural language question about the image as an input and generates natural language answer as an output.

VQA is interesting because it requires combining visual and language understanding. A model that solves this task demonstrates a more general understanding of images: it must be able to answer completely different questions about an image, oftentimes even addressing different sections of the image.



1.2 Sources and useful links

Video Description about the problem statement : https://www.youtube.com/watch?v=ElZADFTer4I&t=1345s (https://www.youtube.com/watch?v=ElZADFTer4I&t=1345s)

Dataset Overview (video) : https://www.youtube.com/watch?v=nMr_sSAMpkE (https://www.youtube.com/watch?v=nMr_sSAMpkE)

Dataset source : https://visualqa.org/download.html (https://visualqa.org/download.html)

Dataset download : https://github.com/GT-Vision-Lab/VQA/blob/master/README.md (https://github.com/GT-Vision-Lab/VQA/blob/master/README.md)

BLOG for reference : https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering/ (https://tryolabs.com/blog/2018/03/01/introduction-to-visual-question-answering/)

RESEARCH PAPERS :

- VQA: Visual Question Answering : https://arxiv.org/abs/1505.00468 (https://arxiv.org/abs/1505.00468)
- Hierarchical Question-Image Co-Attention for Visual Question Answering : https://arxiv.org/abs/1606.00061 (https://arxiv.org/abs/1606.00061)

1.3 Dataset overview

VQA v2.0 release

This release consists of Real images

- 82,783 MS COCO training images, 40,504 MS COCO validation images and 81,434 MS COCO testing images (images are obtained from [MS COCO website] (http://mscoco.org/dataset/#download (http://mscoco.org/dataset/#download)))
- 443,757 questions for training, 214,354 questions for validation and 447,793 questions for testing
- 4,437,570 answers for training and 2,143,540 answers for validation (10 per question)

There is only one type of task for this dataset: Open-ended visual question answering

1.4 Importing the useful libraries

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
import json
import os
import re
import argparse

import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
import numpy as np
import pickle
import time as tm
import seaborn as sns
import plotly.express as px
from tqdm import tqdm
```

1.4 Download the dataset (Questions and Annotations)

In [2]:
```
# Reference : https://github.com/jiasenlu/HieCoAttenVQA/blob/master/data/vqa_preprocess.py
# NOTE :
    # The python version used in the repo is 2.0 and I am using python-3.7.
    # So I have to make some code changes
# Download the VQA Questions from http://www.visualqa.org/download.html
# NOTE : The links are updated. The links in the given file (by the website) are expired

def download_vqa():

    print('Download started....')

    ########## Download the VQA questions ##########

    # Checking if the directory contains the file
    if os.path.isfile('data/v2_Questions_Train_mscoco.zip'):
        print('File already downloaded.... Skipping.....')
    else:
        os.system('wget https://s3.amazonaws.com/cvmlp/vqa/mscoco/vqa/v2_Questions_Train_mscoco.zip -P data/')
        print('Train Questions downloaded...')

    # Checking if the directory contains the file
    if os.path.isfile('data/v2_Questions_Val_mscoco.zip'):
        print('File already downloaded.... Skipping.....')
    else:
        os.system('wget https://s3.amazonaws.com/cvmlp/vqa/mscoco/vqa/v2_Questions_Val_mscoco.zip -P data/')
        print('Validation Questions downloaded...')

    # Checking if the directory contains the file
    if os.path.isfile('data/v2_Questions_Test_mscoco.zip'):
        print('File already downloaded.... Skipping.....')
    else:
        os.system('wget https://s3.amazonaws.com/cvmlp/vqa/mscoco/vqa/v2_Questions_Test_mscoco.zip -P data/')
        print('Test Questions downloaded...')


    ########## Download the VQA Annotations ##########

    # Checking if the directory contains the file
    if os.path.isfile('data/v2_Annotations_Train_mscoco.zip'):
        print('File already downloaded.... Skipping.....')
    else:
        os.system('wget https://s3.amazonaws.com/cvmlp/vqa/mscoco/vqa/v2_Annotations_Train_mscoco.zip -P data/')
        print('Train Annotations downloaded...')

    # Checking if the directory contains the file
    if os.path.isfile('data/v2_Annotations_Val_mscoco.zip'):
        print('File already downloaded.... Skipping.....')
    else:
        os.system('wget https://s3.amazonaws.com/cvmlp/vqa/mscoco/vqa/v2_Annotations_Val_mscoco.zip -P data/')
        print('Validation Annotations downloaded...')


    # Unzip the annotations
    print('Unzipping the datasets...')
    os.system('unzip data/v2_Questions_Train_mscoco.zip -d annotations/')
    os.system('unzip data/v2_Questions_Val_mscoco.zip -d annotations/')
    os.system('unzip data/v2_Questions_Test_mscoco.zip -d annotations/')
    os.system('unzip data/v2_Annotations_Train_mscoco.zip -d annotations/')
    os.system('unzip data/v2_Annotations_Val_mscoco.zip -d annotations/')
    print("..DONE..")
```

In [3]:
```
## Downloading the dataset
download_vqa()
```

```
Download started....
File already downloaded.... Skipping.....
File already downloaded.... Skipping.....
File already downloaded.... Skipping.....
File already downloaded.... Skipping.....
File already downloaded.... Skipping.....
Unzipping the datasets...
..DONE..
```

2.1 Loading the downloaded train_annotations and train_questions json files

In [4]:
```
# Loading the annotations
train_anno = json.load(open('annotations/v2_mscoco_train2014_annotations.json', 'r'))

# Loading the Questions
train_ques = json.load(open('annotations/v2_OpenEnded_mscoco_train2014_questions.json', 'r'))
```

In [5]:
```
train_anno
```

Out[5]:
```
{'info': {'description': 'This is v2.0 of the VQA dataset.',
  'url': 'http://visualqa.org',
  'version': '2.0',
  'year': 2017,
  'contributor': 'VQA Team',
  'date_created': '2017-04-26 17:07:13'},
 'license': {'url': 'http://creativecommons.org/licenses/by/4.0/',
  'name': 'Creative Commons Attribution 4.0 International License'},
 'data_subtype': 'train2014',
 'annotations': [{'question_type': 'what is this',
   'multiple_choice_answer': 'net',
   'answers': [{'answer': 'net', 'answer_confidence': 'maybe', 'answer_id': 1},
    {'answer': 'net', 'answer_confidence': 'yes', 'answer_id': 2},
    {'answer': 'net', 'answer_confidence': 'yes', 'answer_id': 3},
    {'answer': 'netting', 'answer_confidence': 'yes', 'answer_id': 4},
    {'answer': 'net', 'answer_confidence': 'yes', 'answer_id': 5},
    {'answer': 'net', 'answer_confidence': 'yes', 'answer_id': 6},
    {'answer': 'mesh', 'answer_confidence': 'maybe', 'answer_id': 7},
    {'answer': 'net', 'answer_confidence': 'yes', 'answer_id': 8},
```

In [6]:
```
train_ques
```

Out[6]:
```
{'info': {'description': 'This is v2.0 of the VQA dataset.',
  'url': 'http://visualqa.org',
  'version': '2.0',
  'year': 2017,
  'contributor': 'VQA Team',
  'date_created': '2017-04-26 17:07:13'},
 'task_type': 'Open-Ended',
 'data_type': 'mscoco',
 'license': {'url': 'http://creativecommons.org/licenses/by/4.0/',
  'name': 'Creative Commons Attribution 4.0 International License'},
 'data_subtype': 'train2014',
 'questions': [{'image_id': 458752,
   'question': 'What is this photo taken looking through?',
   'question_id': 458752000},
  {'image_id': 458752,
   'question': 'What position is this man playing?',
   'question_id': 458752001},
  {'image_id': 458752,
   'question': 'What color is the players shirt?',
```

NOTE:

1. As we can see that the file contains some metadata about the dataset and along with that annotations and questions for the respective images.
2. We will just pick the `annotations` key from the `train_anno` and `questions` key from the `train_ques`

2.2 Creating new dataframes of Annotations and Questions

In [7]:
```
# Dataframe for annotations
train_anno_df = pd.DataFrame(train_anno['annotations'])
del train_anno

# Dataframe for questions
train_ques_df = pd.DataFrame(train_ques['questions'])
del train_ques

print("Number of Annotations : ", train_anno_df.shape)
print("Number of Questions : ", train_ques_df.shape)
```

```
Number of Annotations :  (443757, 6)
Number of Questions :  (443757, 3)
```

In [8]:
```
train_anno_df.head()
```

Out[8]:

	answer_type	answers	image_id	multiple_choice_answer	question_id	question_type
0	other	[{'answer': 'net', 'answer_confidence': 'maybe...	458752	net	458752000	what is this
1	other	[{'answer': 'pitcher', 'answer_confidence': 'y...	458752	pitcher	458752001	what
2	other	[{'answer': 'orange', 'answer_confidence': 'ye...	458752	orange	458752002	what color is the
3	yes/no	[{'answer': 'yes', 'answer_confidence': 'yes',...	458752	yes	458752003	is this
4	other	[{'answer': 'white', 'answer_confidence': 'yes...	262146	white	262146000	what color is the

2.2.1 Expanding the answers column

In [9]:
```
pd.DataFrame(train_anno_df['answers'][0])
```

Out[9]:

	answer	answer_confidence	answer_id
0	net	maybe	1
1	net	yes	2
2	net	yes	3
3	netting	yes	4
4	net	yes	5
5	net	yes	6
6	mesh	maybe	7
7	net	yes	8
8	net	yes	9
9	net	yes	10

In [10]:
```
train_ques_df.head()
```

Out[10]:

	image_id	question	question_id
0	458752	What is this photo taken looking through?	458752000
1	458752	What position is this man playing?	458752001
2	458752	What color is the players shirt?	458752002
3	458752	Is this man a professional baseball player?	458752003
4	262146	What color is the snow?	262146000

In [11]:
```
train_anno_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 443757 entries, 0 to 443756
Data columns (total 6 columns):
answer_type               443757 non-null object
answers                   443757 non-null object
image_id                  443757 non-null int64
multiple_choice_answer    443757 non-null object
question_id               443757 non-null int64
question_type             443757 non-null object
dtypes: int64(2), object(4)
memory usage: 20.3+ MB
```

In [12]:
```
train_ques_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 443757 entries, 0 to 443756
Data columns (total 3 columns):
image_id       443757 non-null int64
question       443757 non-null object
question_id    443757 non-null int64
dtypes: int64(2), object(1)
memory usage: 10.2+ MB
```

NOTE:

1. There are no null vales in the dataset.
2. The annotations file contains 6 columns.
   - `['answer_type', 'answers', 'image_id', 'multiple_choice_answer', 'question_id', 'question_type']`
3. The questions file contains 3 columns.
   - `['image_id', 'question', 'question_id']`
4. For every question there are 10 answers with an answer_confidence. Averaging the answer confidence the final answer has been decided as the multiple_choice_answer column.

2.3 Creating the train and test splits

NOTE:

1. We will keep the following columns for data analysis and after the data analysis we will remove these columns
   - `['answer_type', 'image_id', 'multiple_choice_answer', 'question_id', 'question_type', 'question']`

In [13]:
```
def split_dataset(split=0):

    '''
    Put the VQA data into single json file,
    where [[Question_id, Image_id, Question, multipleChoice_answer, Answer] ... ]
    '''

    train = []
    test = []
    imdir = 'images/%s/COCO_%s_%012d.jpg'

    if split == 1:

        print('Loading annotations and questions...')

        # Loading the annotations
        train_anno = json.load(open('annotations/v2_mscoco_train2014_annotations.json', 'r'))
        val_anno = json.load(open('annotations/v2_mscoco_val2014_annotations.json', 'r'))

        # Loading the Questions
        train_ques = json.load(open('annotations/v2_OpenEnded_mscoco_train2014_questions.json', 'r'))
        val_ques = json.load(open('annotations/v2_OpenEnded_mscoco_val2014_questions.json', 'r'))

        # Creating a list containing the question_id, image_path, question, ans, ans_type, ques_type
        subtype = 'train2014'
        for i in tqdm(range(len(train_anno['annotations']))):

            ans = train_anno['annotations'][i]['multiple_choice_answer']
            question_id = train_anno['annotations'][i]['question_id']
            image_path = imdir%(subtype, subtype, train_anno['annotations'][i]['image_id'])
            question = train_ques['questions'][i]['question']
            question_type = train_anno['annotations'][i]['question_type']
            ans_type = train_anno['annotations'][i]['answer_type']

            train.append({'ques_id': question_id,
                          'img_path': image_path,
                          'question': question,
                          'ans': ans,
                          'ans_type': ans_type,
                          'ques_type': question_type})

        # Creating a list containing the question_id, image_path, question, ans,  ans_type, ques_type
        subtype = 'val2014'
        for i in tqdm(range(len(val_anno['annotations']))):
            ans = val_anno['annotations'][i]['multiple_choice_answer']
            question_id = val_anno['annotations'][i]['question_id']
            image_path = imdir%(subtype, subtype, val_anno['annotations'][i]['image_id'])
            question = val_ques['questions'][i]['question']
            question_type = train_anno['annotations'][i]['question_type']
            ans_type = train_anno['annotations'][i]['answer_type']

            test.append({'ques_id': question_id,
                         'img_path': image_path,
                         'question': question,
                         'ans': ans,
                         'ans_type': ans_type,
                         'ques_type': question_type})


        print('Training samples : {}, Testing sample : {}'.format(len(train), len(test)))

        json.dump(train, open('data/vqa_raw_train.json', 'w'))
        json.dump(test, open('data/vqa_raw_test.json', 'w'))


## Splitting the dataset into train and test
split_dataset(split=1)
```

```
Loading annotations and questions...

100%|██████████| 443757/443757 [00:01<00:00, 425476.24it/s]
100%|██████████| 214354/214354 [00:00<00:00, 471093.95it/s]

Training samples : 443757, Testing sample : 214354
```

****

3. Performing EDA on the data

3.1 Loading the data

In [14]:
```
# Loading the train data
vqa_train_df = pd.DataFrame(json.load(open('data/vqa_raw_train.json', 'r')))
vqa_train_df.head()
```

Out[14]:

	ans	ans_type	img_path	ques_id	ques_type	question
0	net	other	images/train2014/COCO_train2014_000000458752.jpg	458752000	what is this	What is this photo taken looking through?
1	pitcher	other	images/train2014/COCO_train2014_000000458752.jpg	458752001	what	What position is this man playing?
2	orange	other	images/train2014/COCO_train2014_000000458752.jpg	458752002	what color is the	What color is the players shirt?
3	yes	yes/no	images/train2014/COCO_train2014_000000458752.jpg	458752003	is this	Is this man a professional baseball player?
4	white	other	images/train2014/COCO_train2014_000000262146.jpg	262146000	what color is the	What color is the snow?

In [15]:
```
print("Number of datapoints the dataset :          ", len(vqa_train_df))
print("Number of unique question id :              ", len(vqa_train_df['ques_id'].unique()))
print("Number of unique question types :           ", len(vqa_train_df['ques_type'].unique()))
print("Number of unique questions in the dataset : ", len(vqa_train_df['question'].unique()))
print("Number of unique answers types :            ", len(vqa_train_df['ans_type'].unique()))
print("Number of unique answers :                  ", len(vqa_train_df['ans'].unique()))
print("Number of unique images :                   ", len(vqa_train_df['img_path'].unique()))
```

```
Number of datapoints the dataset :          443757
Number of unique question id :              443757
Number of unique question types :           65
Number of unique questions in the dataset :  152050
Number of unique answers types :            3
Number of unique answers :                  22531
Number of unique images :                   82783
```

3.2 Testing for any NULL columns

In [16]:
```
vqa_train_df[vqa_train_df.isnull().any(axis=1)]
```

Out[16]:

ans	ans_type	img_path	ques_id	ques_type	question

NOTE:

There are no null values in the dataset

****

[X] Univariate analysis for `question` in the dataset

3.3 Preprocessing the question column

In [17]:
```
# %%time
# # Note:
# # This block takes a lot of time to run. So skipping it for now.....

# # Function for preprocessing the questions
# def nlp_preprocessing(total_text, index, column):

#     if type(total_text) is not int:
#         # replace every special char with space
#         total_text = re.sub('[^a-zA-Z0-9\n]', ' ', total_text)
#         # replace multiple spaces with single space
#         total_text = re.sub('\s+',' ', total_text)
#         # converting all the chars into lower-case.
#         total_text = total_text.lower()

#     vqa_train_df[column][index] = total_text

# # Text processing stage.
# for index, qtn in tqdm(vqa_train_df.iterrows()):
#     if type(qtn['question']) is str:
#         nlp_preprocessing(qtn['question'], index, 'question')
#     else:
#         print("there is no text description for id:",index)
```

3.4 Testing for duplicate questions

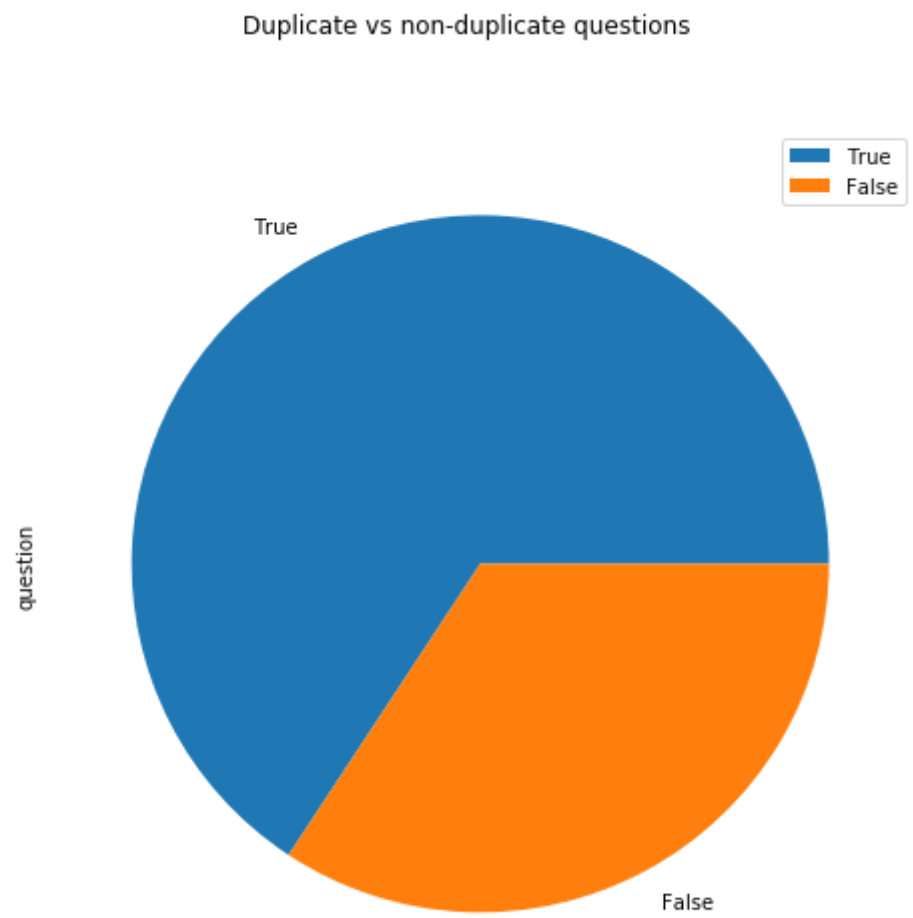
In [18]:
```
vqa_train_df['question'].duplicated().value_counts()
```

Out[18]:
```
True     291707
False    152050
Name: question, dtype: int64
```

3.4.1 Plotting the distribution of duplicate and non duplicate questions

In [19]:
```
fig, ax = plt.subplots(figsize=(8,8))
pd.DataFrame(vqa_train_df['question'].duplicated().value_counts()).plot(kind='pie',
                                                                        subplots=True,
                                                                        ax=ax,
                                                                        title='Duplicate vs non-duplicate questions')
```

Out[19]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0a6aecc358>],
      dtype=object)
```



NOTE :

1. Out of 443757 training examples, 291707 contains duplicate questions.
2. Most of the questions are repeated for some different images because there are 82,783 images in the training dataset and for each image there are multiple questions.
3. This means there are similar types of images as well to which the same questions are being asked. This is why we get so many repeated questions.

3.4.2 Let's see how many questions are there per question type

In [20]:
```
# Creating a dataframe with counts of unique question_ids
# This will give us an idea about how many questions are repeating as per the question type
# and what kind of questions are categorized under which question type

pd.DataFrame(vqa_train_df.groupby(['ques_type','question'])['question'].count())
```

Out[20]:

ques_type	question	question
are	Are this colored roses?	2
	Are this man's friends invited to the party?	2
	Are 2 animals standing in a field?	2
	Are 2 or more people more than likely going on this trip?	2
	Are Arabic words on the truck?	2
	Are Chinese beers popular?	1
	Are Gunning and Gundaroo the same direction?	1
	Are How many trees in the photo?	2
	Are Is this luggage scuffed up?	2
	Are U-turns allowed at this light?	1
	Are a lot of people flying kites in the same area as the girl?	2
	Are a lot of people waiting to get on?	2
	Are adults or children playing?	2
	Are aircrafts essential to human migration?	1
	Are alien lizard people living under the sand?	1
	Are all 3 men wearing the same color?	1
	Are all 3 of the animals in these pictures the same species?	2
	Are all 3 of these images of the same surfer?	1
	Are all 4 wheels on the ground?	2
	Are all 4 wheels the same color?	2
	Are all animals of the same breed?	2
	Are all arrows green?	2
	Are all birds facing the same direction?	1
	Are all buses double-deckers?	2
	Are all cars clearly in focus?	2
	Are all cars going in the same direction?	2
	Are all cats of the same breed?	2
	Are all cattle facing the camera?	2
	Are all chairs alike?	2
	Are all clocks showing the same time?	2
...	...	...
why is the	Why is the woman using an umbrella?	2
	Why is the woman using the umbrella?	2
	Why is the woman walking in front of the vehicles?	2
	Why is the woman wearing a bracelet?	1
	Why is the woman wearing a coat?	2
	Why is the woman wearing a crown?	1
	Why is the woman wearing a helmet?	4
	Why is the woman wearing a scarf?	1
	Why is the woman wearing a veil?	2
	Why is the woman wearing black clothes?	1
	Why is the woman wearing boots?	2
	Why is the woman wearing gloves?	2
	Why is the woman wiping her face?	1
	Why is the woman's face covered?	2
	Why is the woman's hand in the air?	1
	Why is the woman's head blurry?	2
	Why is the women sitting in the fridge?	1
	Why is the word bus upside down?	1
	Why is the wrap red?	1
	Why is the writing upside down?	2
	Why is the yellow sauce placed on the side of the sandwich?	2
	Why is the yellow stripe in the road?	1
	Why is the young man hang in mid air?	1

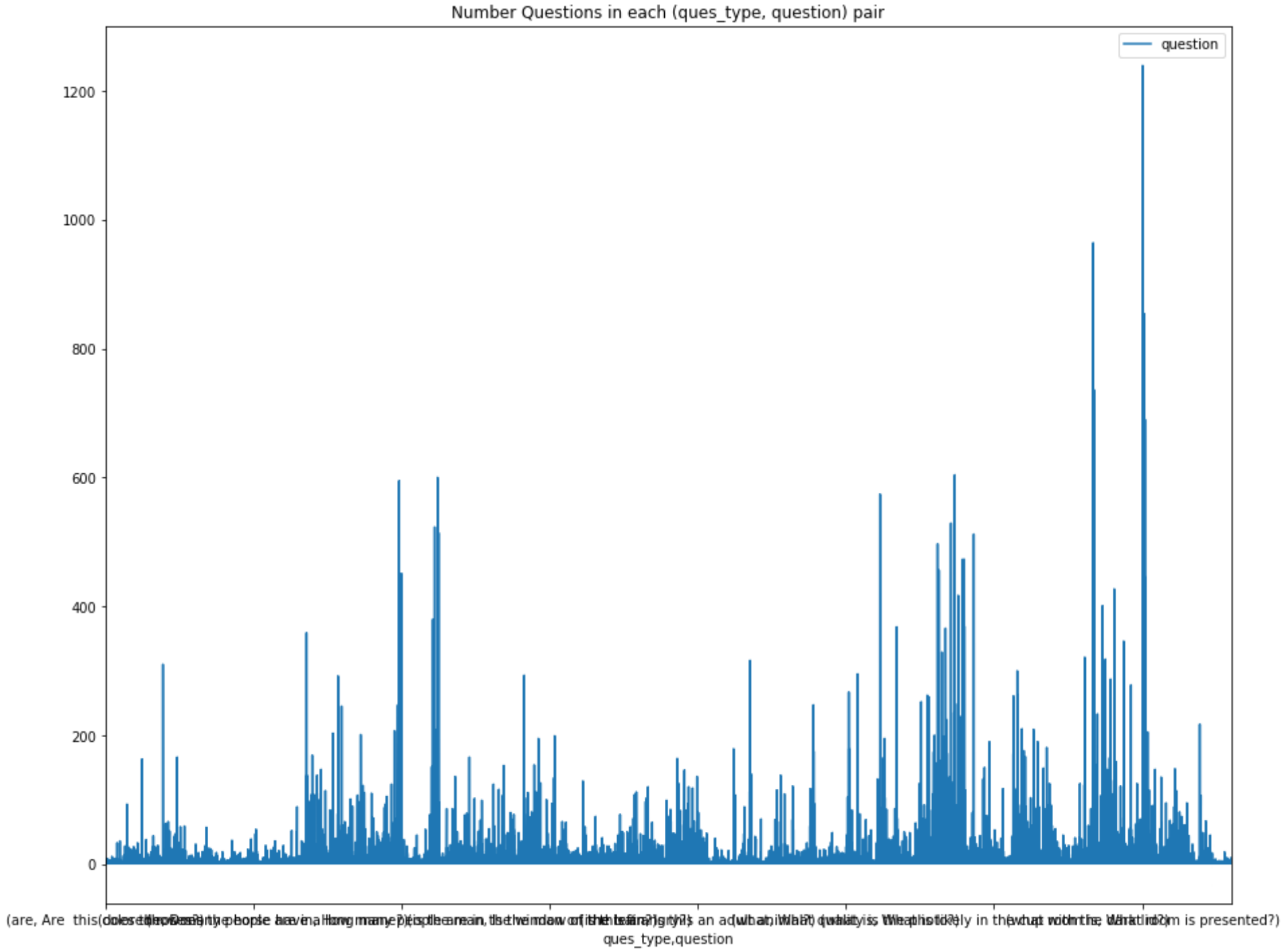
	question
ques_type	**question**
Why is the young man in green shirt and white shorts have his back foot off the ground?	2
Why is the young man in the air?	1
Why is the young woman holding a skateboard?	1
Why is the zebra alone?	2
Why is the zebra doing what it is doing?	2
Why is the zebra there?	2
Why is the zebras tail curled?	2

152050 rows × 1 columns

3.4.3 Plotting the frequencies of the repeating questions

In [21]:
```
# In the block the frequencies of the pair of (question_type , question_id) is plotted.
# This means we can see what is the frequency of each of the pair of the (question_type , question_id)
# i.e how many times that question is repeated.

fig, ax = plt.subplots(figsize=(15,12))
pd.DataFrame(vqa_train_df.groupby(['ques_type','question'])['question'].count()).plot(kind='line',
                                                                                      ax=ax,
                                                                                      title='Number Questions in each
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0a3e4e22b0>



NOTE:

1. Since it is difficult to inference anything from this graph so we will use other techniques to inference a little more

3.4.4 Taking the sum of all the question in that specific question_type

In [22]:
```
# This gives the total number of questions per question type.
# There are total 56 question types and summing the count for each question type gives us an estimate
# of the frequency of each question type. (Number of questions in each question type)

pd.DataFrame(vqa_train_df.groupby('question')['ques_type'].value_counts().unstack().sum())
```

Out[22]:

ques_type	0
are	4912.0
are the	10701.0
are there	5877.0
are there any	2790.0
are these	5782.0
are they	3074.0
can you	1728.0
could	1698.0
do	3012.0
do you	1971.0
does the	6103.0
does this	4396.0
has	1827.0
how	4740.0
how many	42339.0
how many people are	4276.0
how many people are in	2071.0
is	6079.0
is he	2534.0
is it	7345.0
is that a	1585.0
is the	34927.0
is the man	4972.0
is the person	1694.0
is the woman	1938.0
is there	6513.0
is there a	9982.0
is this	16444.0
is this a	16024.0
is this an	1981.0
...	...
what are	3277.0
what are the	7225.0
what brand	1600.0
what color	3032.0
what color are the	6183.0
what color is	2649.0
what color is the	27962.0
what does the	4075.0
what is	13561.0
what is in the	3990.0
what is on the	4254.0
what is the	24502.0
what is the color of the	1750.0
what is the man	5238.0
what is the name	1618.0
what is the person	1729.0
what is the woman	1706.0
what is this	3970.0
what kind of	11192.0
what number is	1668.0
what room is	1647.0
what sport is	2527.0
what time	2914.0

ques_type	0
what type of	7962.0
where are the	2161.0
where is the	6734.0
which	5382.0
who is	2154.0
why	3347.0
why is the	1544.0

65 rows × 1 columns

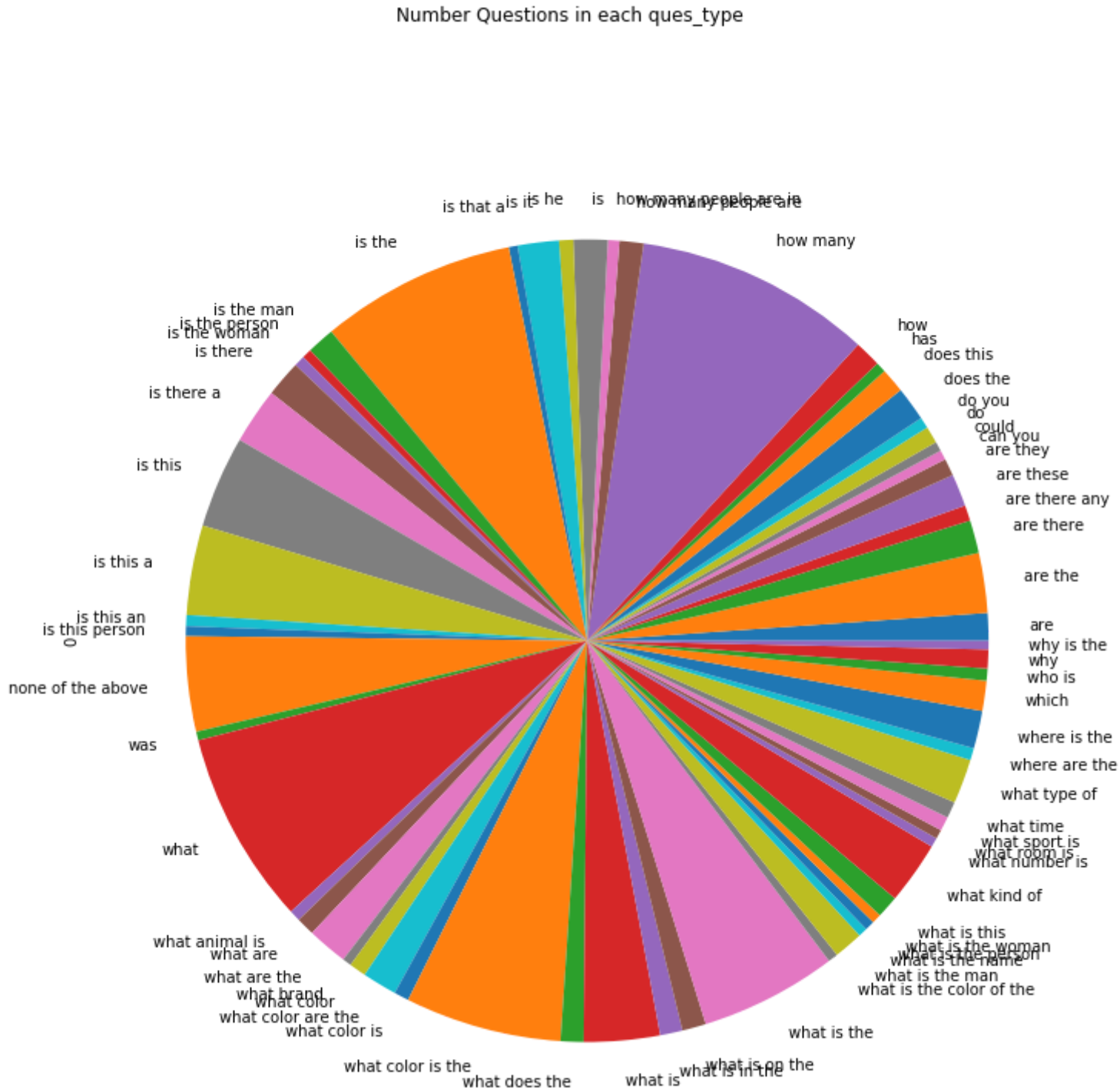
NOTE:

1. We have total 65 question_types and we have the count of all the questions in individual question type.
2. Let's see how they are distributed and which question type has the most number of questions.

3.4.5 Pie chart of questions type distribution for each questions

In [23]:
```
(figsize=(12,12))
_df.groupby('question')['ques_type'].value_counts().unstack().sum()).plot(kind='pie',
                                                                          ax=ax,
                                                                          subplots=True,
                                                                          legend=False,
                                                                          title='Number Questions in each ques_type')
```

Out[23]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0b6483a160>],
      dtype=object)
```



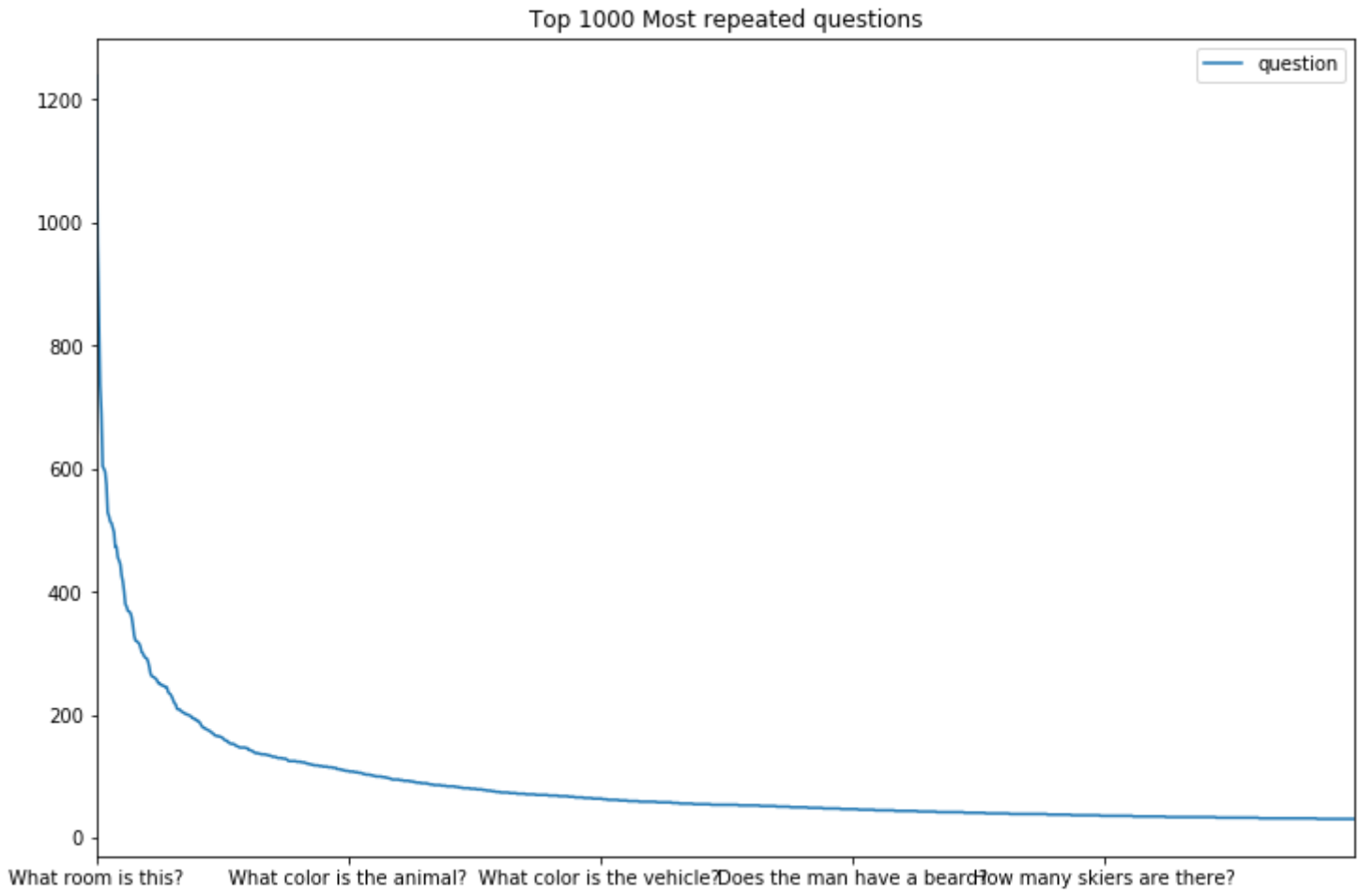
3.4.6 Counting the repeating questions

In [24]:
```
# Counts of the individual questions
# This gives us an idea about how many times this question has been asked.
# Just for demo I am taking only 100 questions
vqa_train_df['question'].value_counts()[0:50,]
```

Out[24]:
```
What room is this?                      1239
What is the man doing?                   964
What sport is this?                      854
What is the man holding?                 736
What time is it?                         690
What color is the plate?                 604
Is it raining?                           600
How many people are there?               595
What animal is this?                     574
What color is the man's shirt?           529
Is it daytime?                           523
Is it sunny?                             514
What does the sign say?                  512
What sport is being played?              505
What color is the bus?                   497
What color is the wall?                  473
What color is the train?                 473
What color is the cat?                   456
How many people are in the picture?      451
What time of day is it?                  445
What kind of animal is this?             427
What color is the sky?                   417
What is the person doing?                401
Is it a sunny day?                       380
How many people are in the photo?        376
What color is the water?                 369
What are the people doing?               368
What color is the grass?                 366
How many animals are there?              359
What kind of food is this?               346
What color is the dog?                   329
What is the weather like?                321
What color is the umbrella?              320
What is the woman holding?               318
Where is this?                           316
Are there clouds in the sky?             310
What color is the truck?                 302
What is in the picture?                  300
What sport are they playing?             295
Is the sky clear?                        293
How many giraffes are there?             292
What is this person doing?               287
What kind of room is this?               278
What season is it?                       267
What is in the sky?                      263
What color are the walls?                262
What is in the background?               261
What color is his shirt?                 259
What is this person holding?             257
What color are the flowers?              252
Name: question, dtype: int64
```

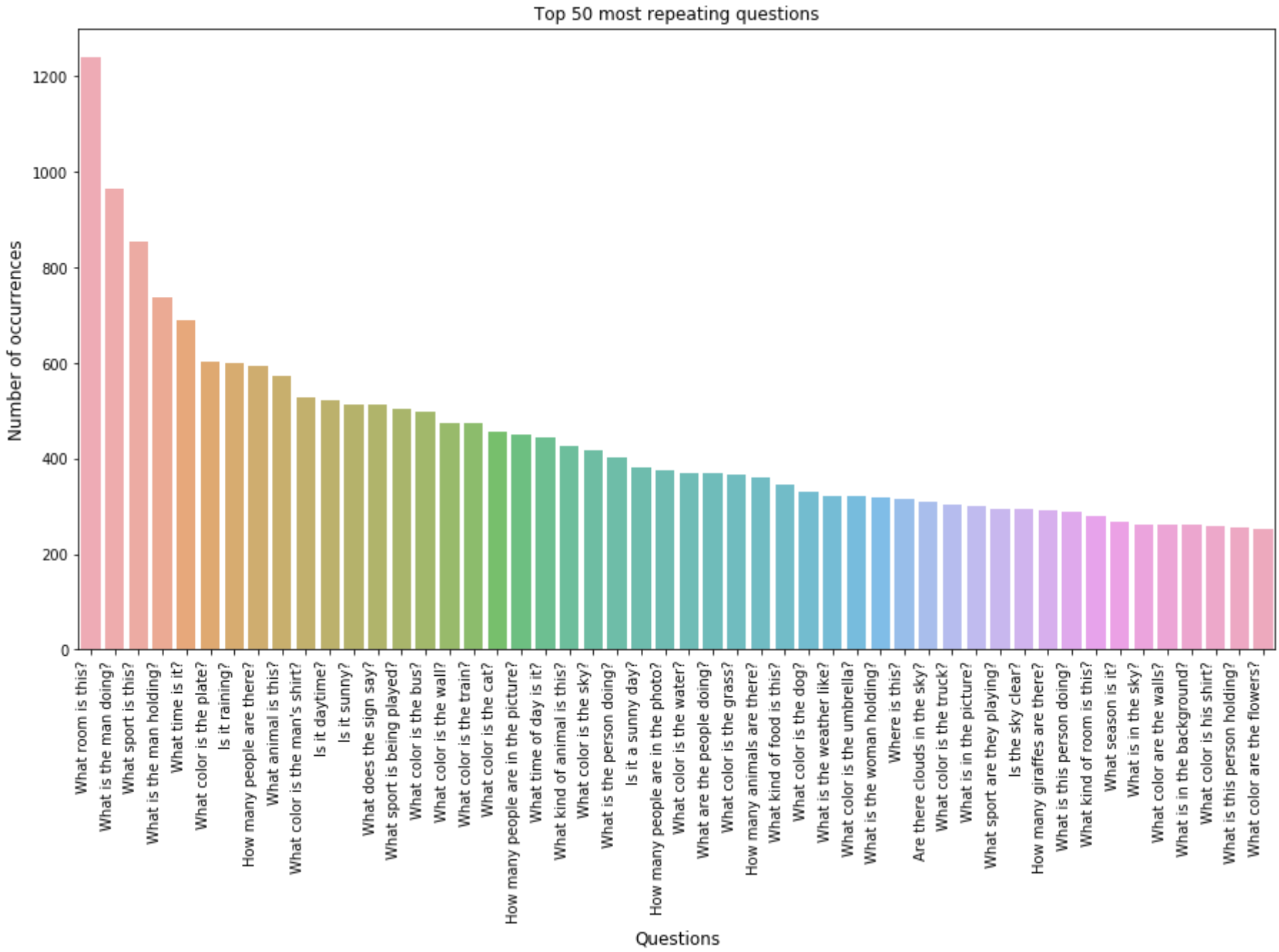
In [25]:
```
# Taking only the top 1000 most repeated question

fig, ax = plt.subplots(figsize=(12,8))
pd.DataFrame(vqa_train_df['question'].value_counts()[0:1000,]).plot(kind='line',
                                                                    ax=ax,
                                                                    legend=True,
                                                                    title='Top 1000 Most repeated questions')
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0b5945ebe0>



3.4.7 Plotting the top 50 most repeating questions

In [26]:
```
qtn_count = vqa_train_df['question'].value_counts()[:50,]

plt.figure(figsize=(15,8))
sns.barplot(qtn_count.index, qtn_count.values, alpha=0.8)
plt.xticks(rotation=90, horizontalalignment='right')
plt.title('Top 50 most repeating questions')
plt.ylabel('Number of occurrences', fontsize=12)
plt.xlabel('Questions', fontsize=12)
plt.show()
```



3.5 Let's plot the distribution of word length in the questions

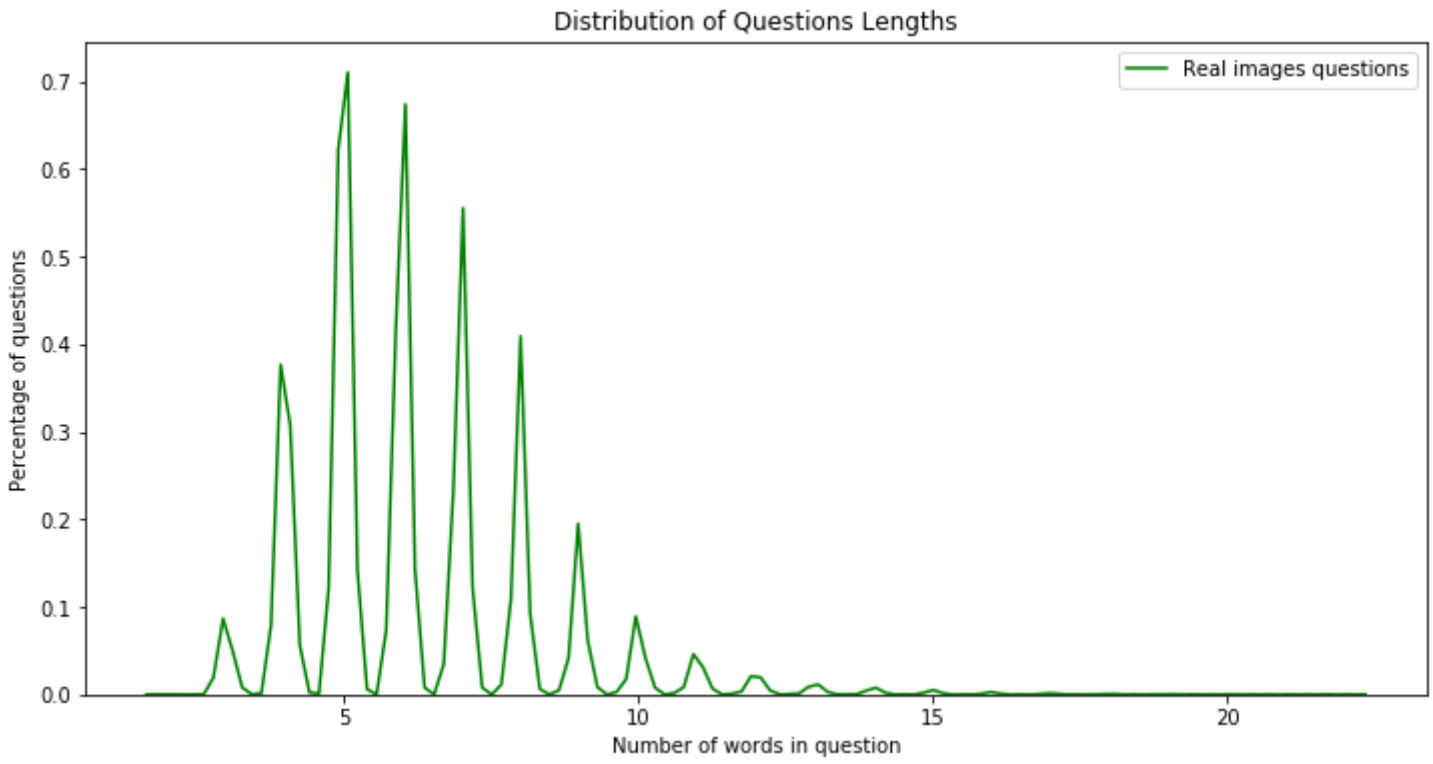
In [27]:
```
## Creating a new column as number of words present in the question column
vqa_train_df["num_words_in_qtns"] = vqa_train_df['question'].apply(lambda x: len(str(x).split()))
```

In [28]:
```
vqa_train_df.head()
```

Out[28]:

	ans	ans_type	img_path	ques_id	ques_type	question	num_words_in_qtns
0	net	other	images/train2014/COCO_train2014_000000458752.jpg	458752000	what is this	What is this photo taken looking through?	7
1	pitcher	other	images/train2014/COCO_train2014_000000458752.jpg	458752001	what	What position is this man playing?	6
2	orange	other	images/train2014/COCO_train2014_000000458752.jpg	458752002	what color is the	What color is the players shirt?	6
3	yes	yes/no	images/train2014/COCO_train2014_000000458752.jpg	458752003	is this	Is this man a professional baseball player?	7
4	white	other	images/train2014/COCO_train2014_000000262146.jpg	262146000	what color is the	What color is the snow?	5

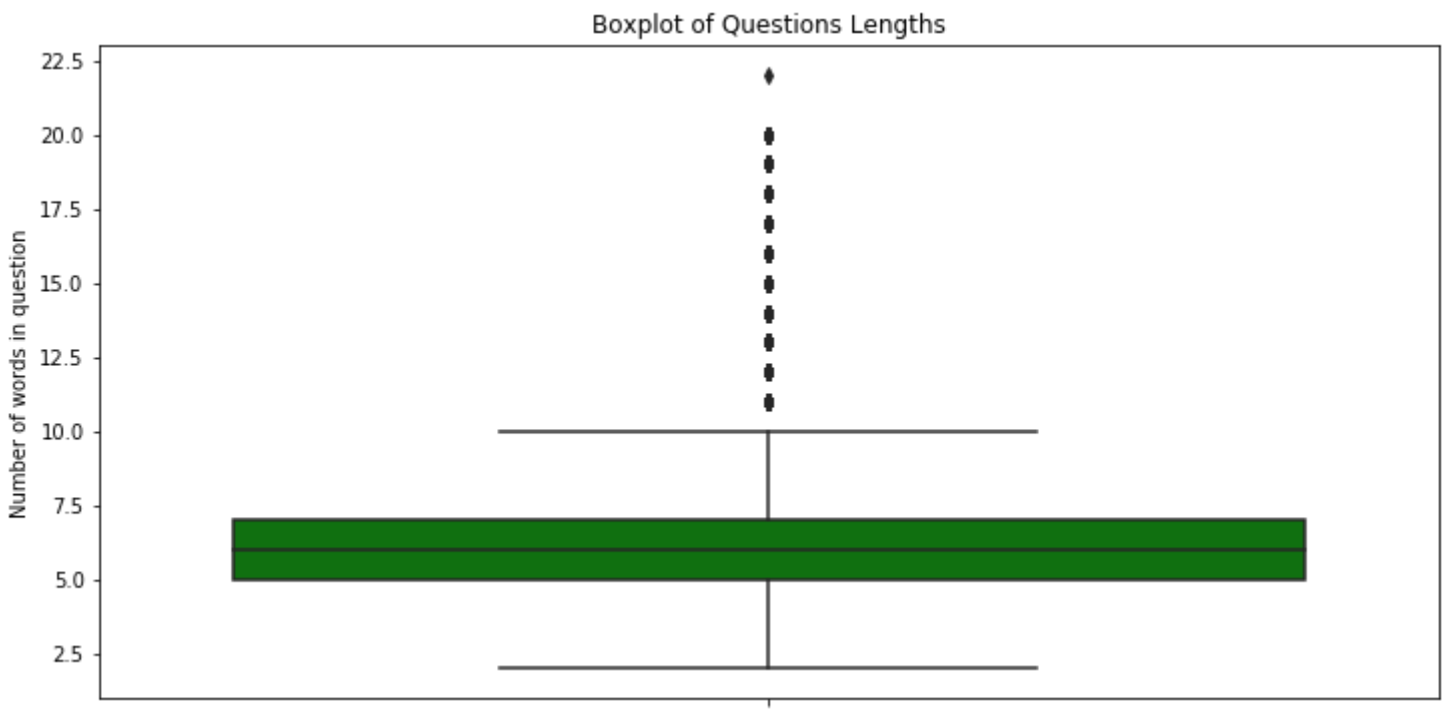
In [29]:
```
# Distribution of question length
plt.figure(figsize=(12,6))
sns.distplot(vqa_train_df['num_words_in_qtns'], hist=False, label='Real images questions', color='g')
plt.title("Distribution of Questions Lengths")
plt.xlabel("Number of words in question")
plt.ylabel("Percentage of questions")
plt.legend()
plt.show()
```



In [30]:
```
print("Maximum length of questions : ", max(vqa_train_df['num_words_in_qtns']))
print("Minumum length of questions : ", min(vqa_train_df['num_words_in_qtns']))
print("Mean length of questions : ", np.mean(vqa_train_df['num_words_in_qtns']))
```

```
Maximum length of questions :  22
Minumum length of questions :  2
Mean length of questions :  6.2015427362272595
```

In [31]:
```
# Boxplot
plt.figure(figsize=(12,6))
sns.boxplot(y=vqa_train_df['num_words_in_qtns'], color='g')
plt.title("Boxplot of Questions Lengths")
plt.ylabel("Number of words in question")
plt.show()
```



NOTE :

1. As we can see from the PDF that most of the questions are of length 5 words (more than 70%)
2. There are very few questions with few number of words or very large number of words.
3. Most questions range from 5 to 7.5 words

3.6 Word cloud will show the most frequent words in the question corpus.

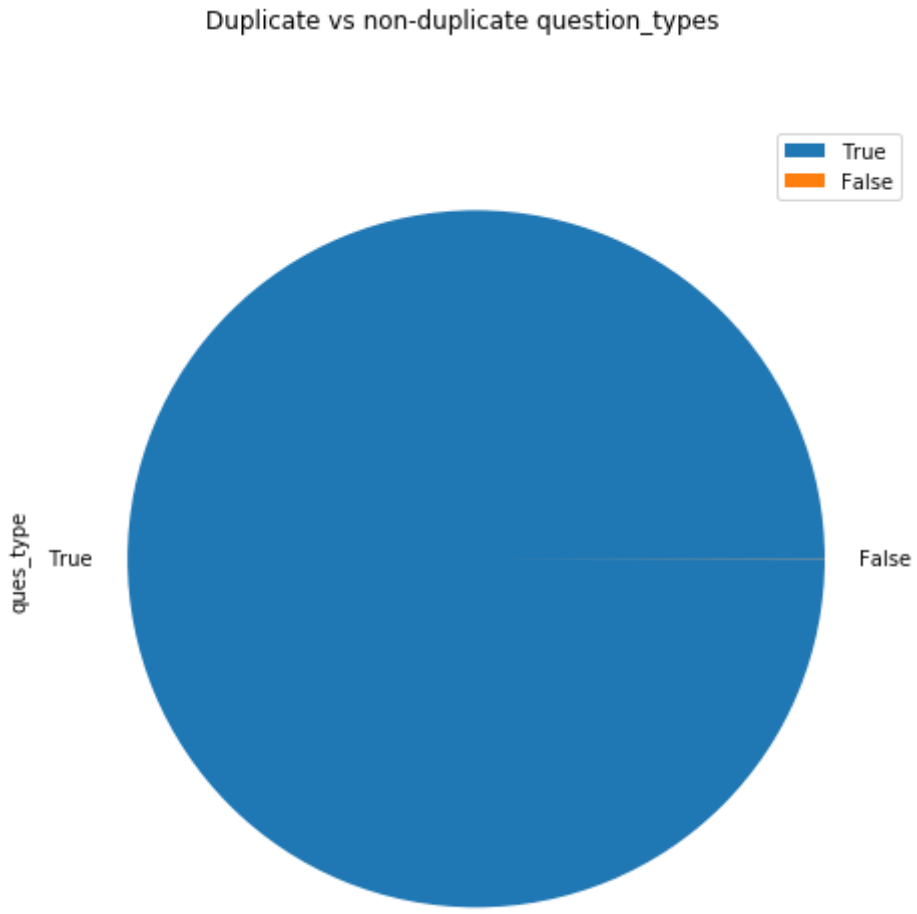
In [32]:
```
# Function to plot the word cloud
def plot_word_cloud(txt):

    cloud = " ".join(word for word in txt)

    # call built-in method WordCloud for creating an object for drawing a word cloud
    wordcloud = WordCloud(width = 1000, height = 500, background_color ='black',
                          stopwords = STOPWORDS).generate(cloud)

    # plot the WordCloud image
    plt.figure(figsize=(30,15))
    plt.imshow(wordcloud, interpolation = 'bilinear')
    plt.show()
```

In [33]:
```
# Plotting the wordcloud
plot_word_cloud(list(vqa_train_df.question.values))
```

3.7 Plotting the distribution of duplicate and non duplicate question types

In [34]:
```
fig, ax = plt.subplots(figsize=(8,8))
pd.DataFrame(vqa_train_df['ques_type'].duplicated().value_counts()).plot(kind = 'pie',
                                                                         subplots=True,
                                                                         ax=ax,
                                                                         title='Duplicate vs non-duplicate question_t
```

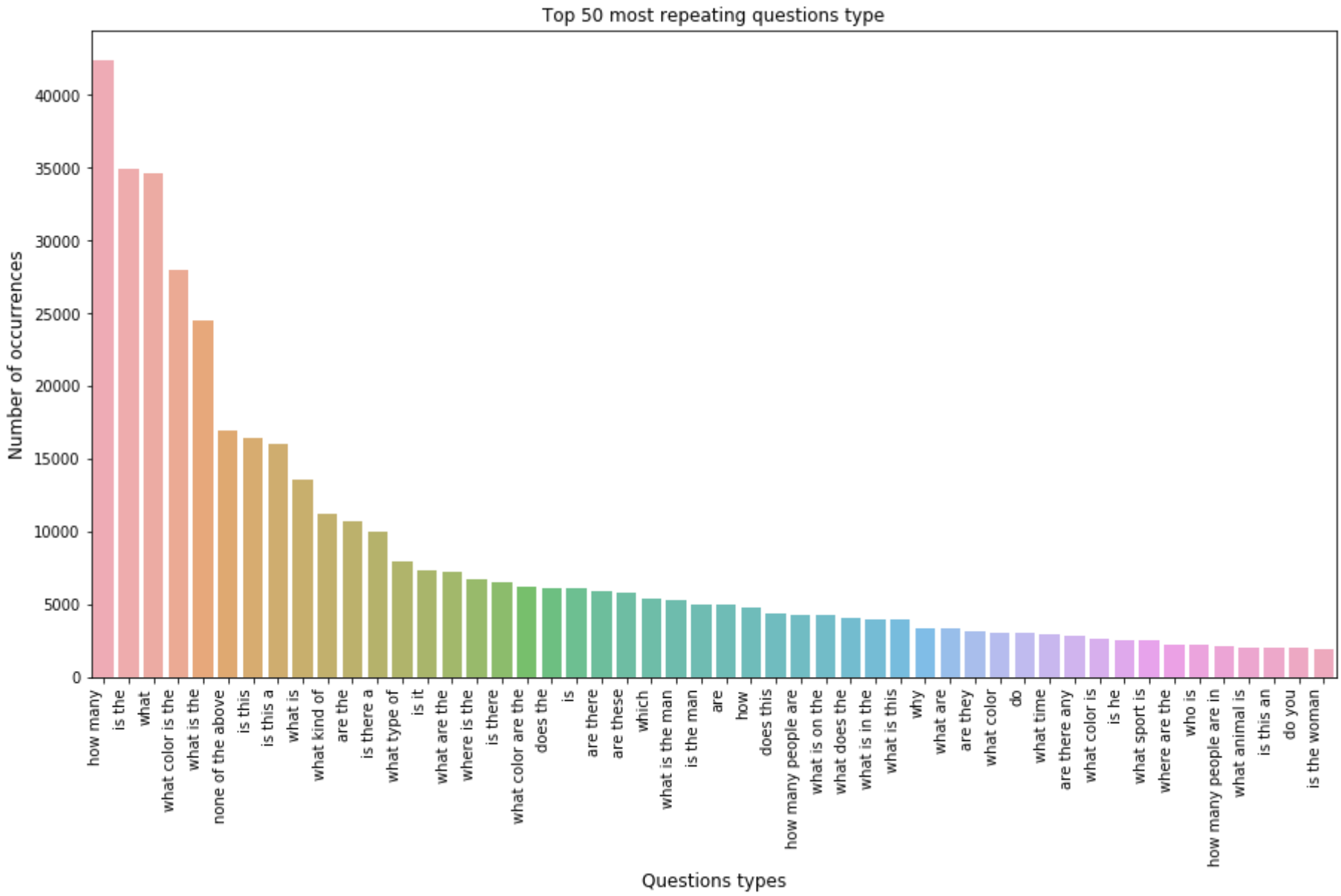
Out[34]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0af21d50b8>],
      dtype=object)
```



NOTE : This shows that almost all the questions are starting with one of the given starting phrase except for very very few questions with unique question_type

3.8 Plotting top 50 most repeating starting phrase

In [35]:
```
qtn_types_counts = vqa_train_df['ques_type'].value_counts()[:50,]

plt.figure(figsize=(15,8))
sns.barplot(qtn_types_counts.index, qtn_types_counts.values, alpha=0.8)
plt.xticks(rotation=90, horizontalalignment='right')
plt.title('Top 50 most repeating questions type')
plt.ylabel('Number of occurrences', fontsize=12)
plt.xlabel('Questions types', fontsize=12)
plt.show()
```



3.9 Combining the question type and the answer type and counting the answers per question type

In [36]:
```
# This will give us a dataframe with all the answer_types categories and a count of the answer
# types per question_type.
# With this we can inference how different question_types are answered.
# Whether they are answered by a YES/NO or With a numerical value or some unique answer (categorized as other)

vqa_train_df.groupby(['ques_type', 'ans_type']).count()['ans'].unstack()
```

Out[36]:

ans_type	number	other	yes/no
ques_type			
are	10.0	146.0	4756.0
are the	2.0	775.0	9924.0
are there	17.0	105.0	5755.0
are there any	1.0	NaN	2789.0
are these	NaN	356.0	5426.0
are they	1.0	138.0	2935.0
can you	9.0	50.0	1669.0
could	NaN	1.0	1697.0
do	3.0	67.0	2942.0
do you	1.0	47.0	1923.0
does the	4.0	237.0	5862.0
does this	2.0	176.0	4218.0
has	1.0	10.0	1816.0
how	1596.0	3144.0	NaN
how many	42125.0	214.0	NaN
how many people are	4257.0	19.0	NaN
how many people are in	2058.0	13.0	NaN
is	1.0	276.0	5802.0
is he	NaN	111.0	2423.0
is it	2.0	752.0	6591.0
is that a	NaN	196.0	1389.0
is the	13.0	3329.0	31585.0
is the man	1.0	272.0	4699.0
is the person	NaN	265.0	1429.0
is the woman	4.0	59.0	1875.0
is there	15.0	45.0	6453.0
is there a	NaN	44.0	9938.0
is this	9.0	1286.0	15149.0
is this a	8.0	1218.0	14798.0
is this an	NaN	147.0	1834.0
...	...	...	...
what are	5.0	3272.0	NaN
what are the	145.0	7080.0	NaN
what brand	4.0	1596.0	NaN
what color	NaN	3032.0	NaN
what color are the	NaN	6183.0	NaN
what color is	NaN	2649.0	NaN
what color is the	NaN	27962.0	NaN
what does the	175.0	3898.0	2.0
what is	85.0	13469.0	7.0
what is in the	NaN	3990.0	NaN
what is on the	3.0	4251.0	NaN
what is the	1459.0	23036.0	7.0
what is the color of the	NaN	1750.0	NaN
what is the man	NaN	5236.0	2.0
what is the name	18.0	1600.0	NaN
what is the person	NaN	1729.0	NaN
what is the woman	NaN	1706.0	NaN
what is this	5.0	3965.0	NaN
what kind of	8.0	11184.0	NaN
what number is	1632.0	36.0	NaN
what room is	NaN	1647.0	NaN
what sport is	NaN	2527.0	NaN

ans_type	number	other	yes/no
ques_type			
what time	1709.0	1205.0	NaN
what type of	8.0	7954.0	NaN
where are the	NaN	2161.0	NaN
where is the	5.0	6729.0	NaN
which	163.0	5219.0	NaN
who is	16.0	2138.0	NaN
why	5.0	3337.0	5.0
why is the	3.0	1541.0	NaN

65 rows × 3 columns

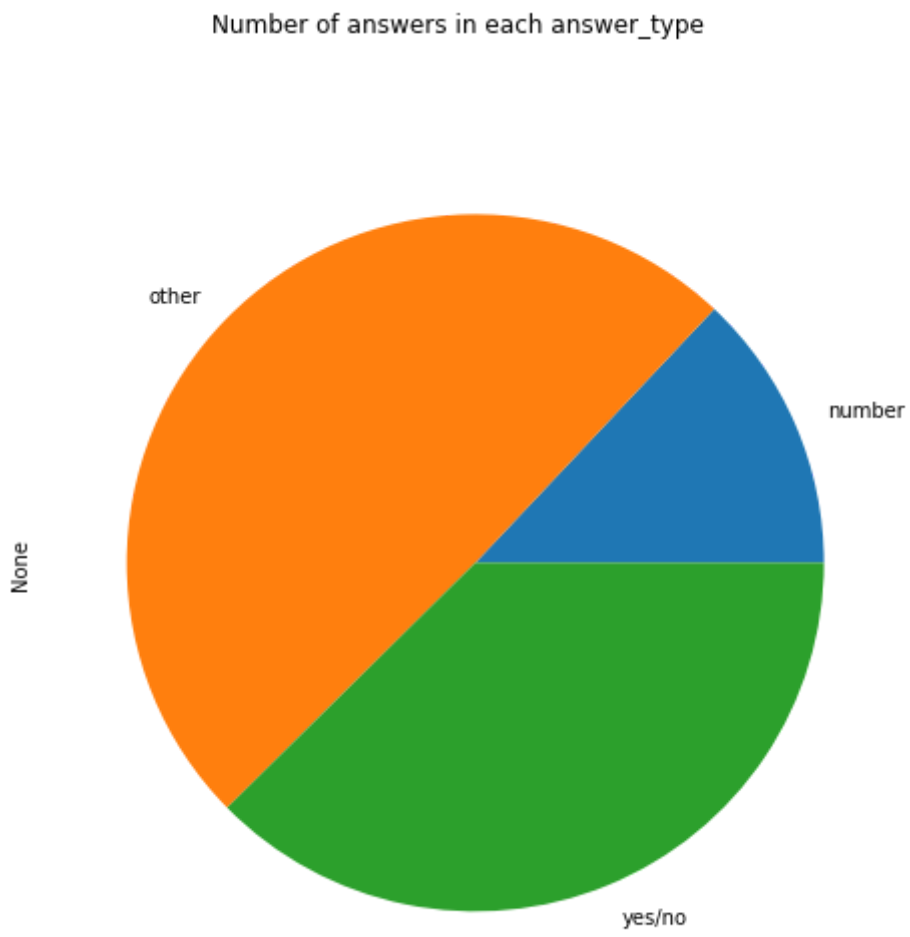
Lets take the sum of all the three columns

In [37]:
```
vqa_train_df.groupby(['ques_type', 'ans_type']).count()['ans'].unstack().sum()
```

Out[37]:
```
ans_type
number     57606.0
other     219269.0
yes/no    166882.0
dtype: float64
```

In [38]:
```
# Plot to understand the distribution of answers in the dataset as per answer type
fig, ax = plt.subplots(figsize=(10,8))
vqa_train_df.groupby(['ques_type', 'ans_type']).count()['ans'].unstack().sum().plot(kind='pie',
                                                                                    ax=ax,
                                                                                    subplots=True,
                                                                                    title='Number of answers ir
```

Out[38]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0add0f5cf8>],
      dtype=object)
```



NOTE:

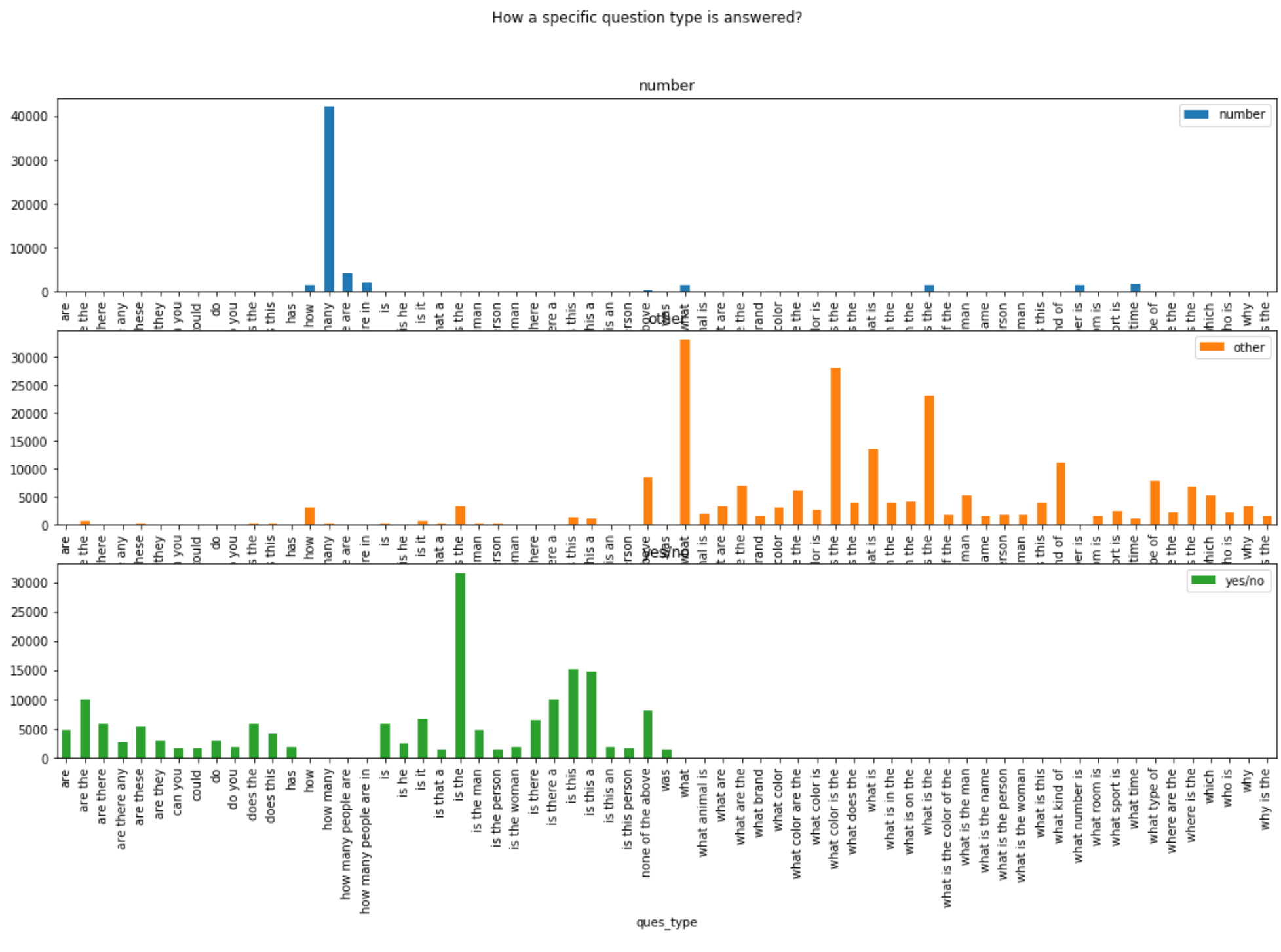
1. As we can see there are 65 questions categories and 3 answer categories.
2. Out of the 3 categories in the answers "0THERS" have the highest number of answers. This means most of the questions have a unique answer to the question asked with respect to the image.
3. If we make this a classification task there will be so many classes to classify from and the image might not always be the same. Same goes for the question. So making it a multiclass classification task will be pretty difficult.

Q. What kind of questions are generally answers by these 3 categories??

3.10 Let's see what kind of answers are answered by a specific type of question

In [39]:
```
# Here we are trying to understand what kind of answers are generally given to a question type.
# By this we will be able to understand how to answer a specific question if that question falls in a specific
# category of question (question_type)

fig, ax = plt.subplots(figsize=(18,10))
vqa_train_df.groupby(['ques_type', 'ans_type']).count()['ans'].unstack().plot(kind='bar',
                                                                              ax=ax,
                                                                              subplots=True,
                                                                              title='How a specific question type is
```

Out[39]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0af21b2780>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f0ae11f4cf8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f0ae7943828>],
      dtype=object)
```

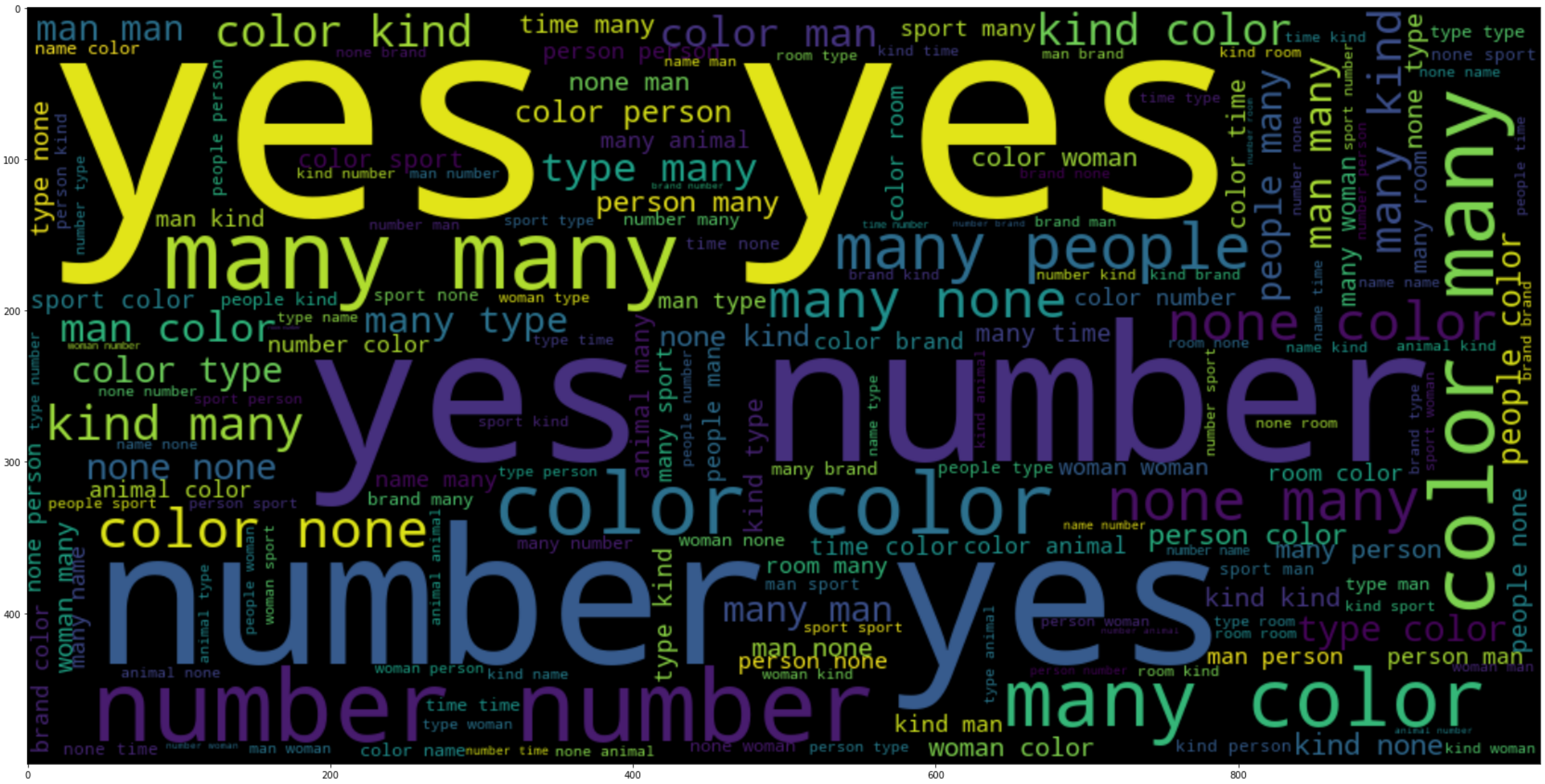


NOTE:

1. We can clearly see that more than 40K questions start with `"How many.."`. That means most answers for these questions will be a numerical value.
2. For `"What.."` kind of questions the answers will be more diverse and will increase the diversity of the answers. As we can see from the plot that most of the questions starting with `"What"` have "OTHER" answer category.
3. Questions starting with `"Is the.."`, `"Are.."`, `"Does.."` etc are typically answered using Yes/No.
4. Other questions types like `"What color"` or `"Which"` have more speciallized responses, such as colors, or "left" and "right", so they have a more diverse answer.

3.10 Plotting the wordcloud for question type + answer type

```
In [40]: # Plotting the wordcloud
plot_word_cloud(list(vqa_train_df.ques_type.values)+list(vqa_train_df.ans_type.values))
```



3.11 Let's plot the distribution of word length in the answers

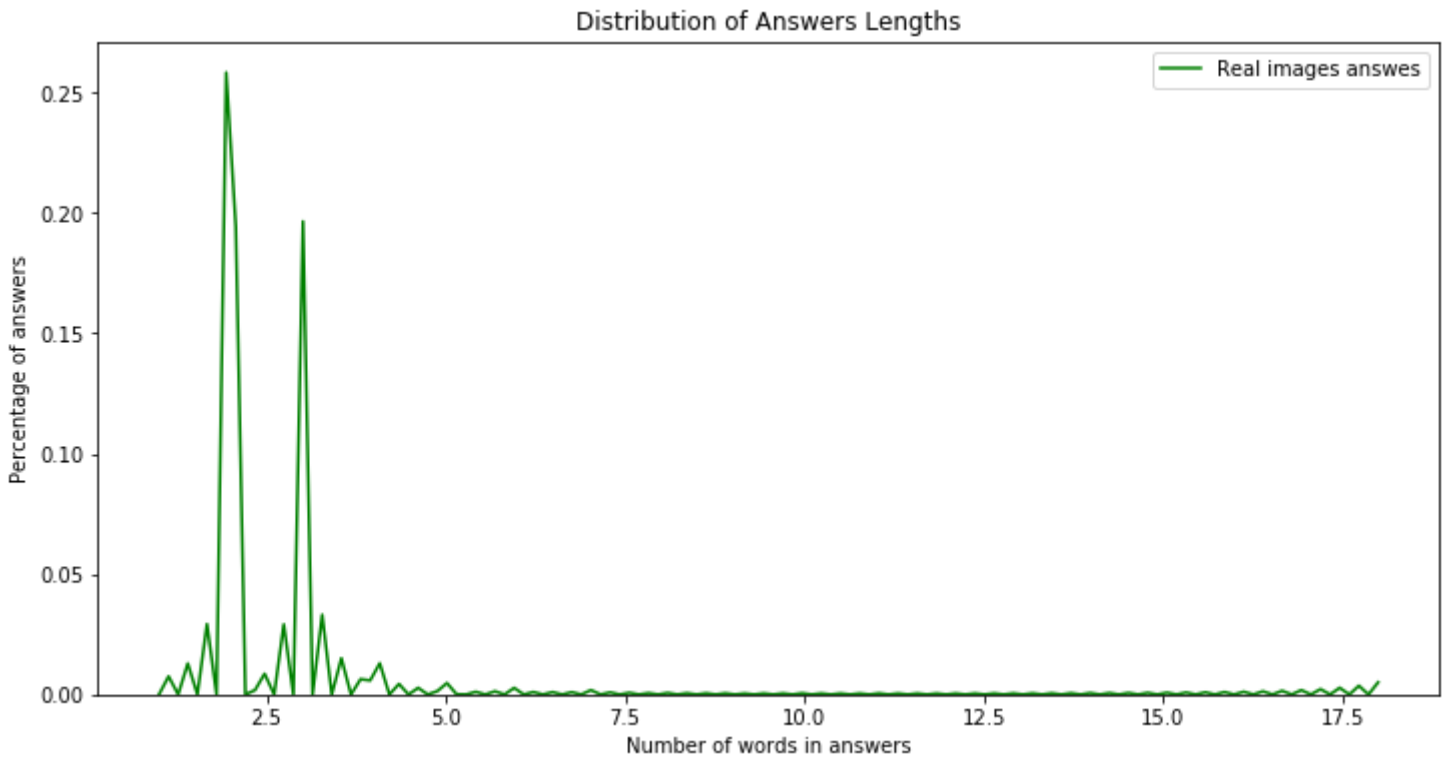
```
In [41]: ## Creating a new column as number of words present in the answer column
vqa_train_df["num_words_in_ans"] = vqa_train_df['ans'].apply(lambda x: len(str(x).split()))
```

```
In [42]: vqa_train_df.head()
```

Out[42]:

	ans	ans_type	img_path	ques_id	ques_type	question	num_words_in_qtns	num_words_in_ans
0	net	other	images/train2014/COCO_train2014_000000458752.jpg	458752000	what is this	What is this photo taken looking through?	7	1
1	pitcher	other	images/train2014/COCO_train2014_000000458752.jpg	458752001	what	What position is this man playing?	6	1
2	orange	other	images/train2014/COCO_train2014_000000458752.jpg	458752002	what color is the	What color is the players shirt?	6	1
3	yes	yes/no	images/train2014/COCO_train2014_000000458752.jpg	458752003	is this	Is this man a professional baseball player?	7	1
4	white	other	images/train2014/COCO_train2014_000000262146.jpg	262146000	what color is the	What color is the snow?	5	1

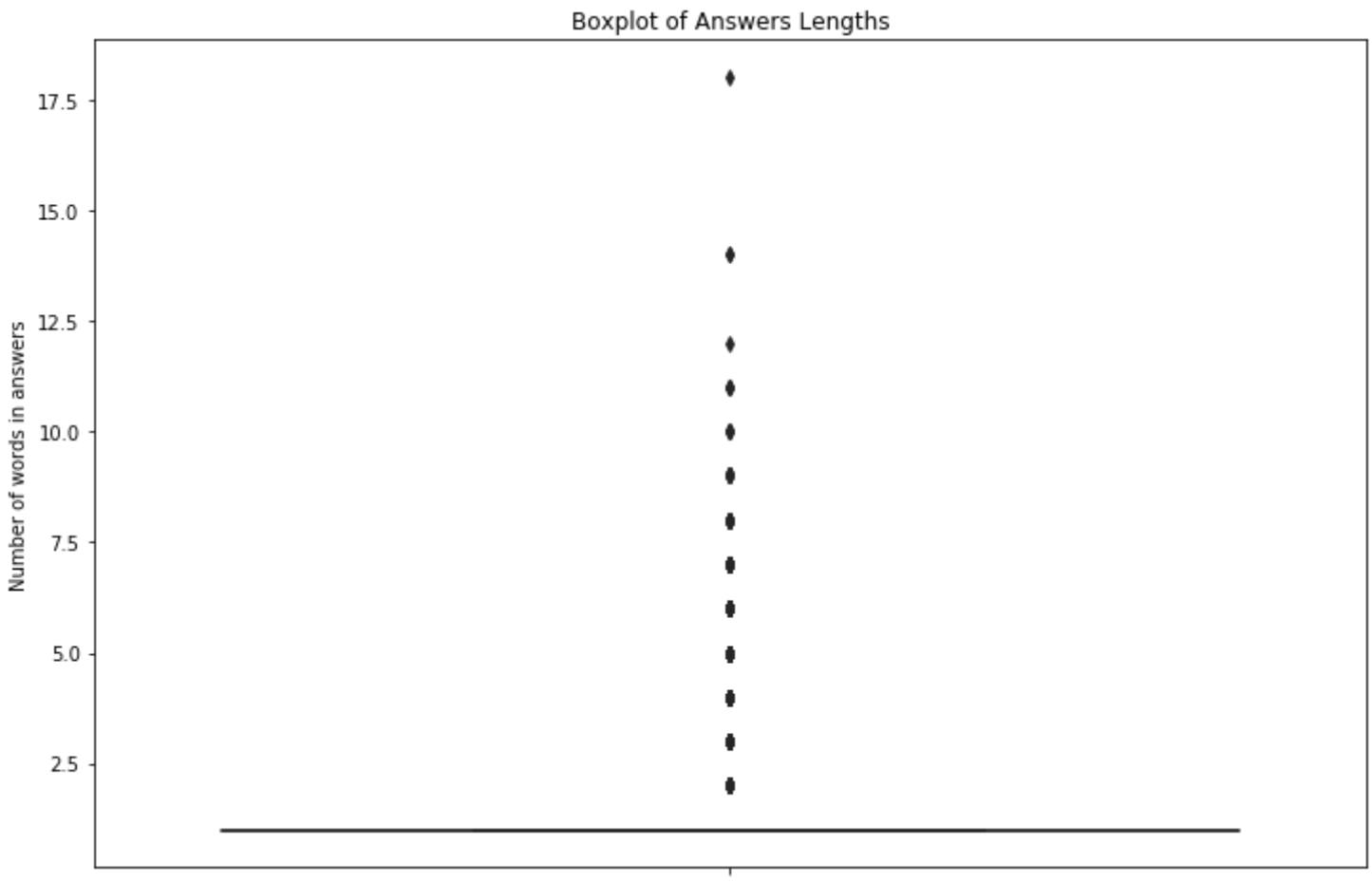
In [43]:
```
# Distribution of ans length
plt.figure(figsize=(12,6))
sns.distplot(vqa_train_df['num_words_in_ans'], hist=False, label='Real images answes', color='g')
plt.title("Distribution of Answers Lengths")
plt.xlabel("Number of words in answers")
plt.ylabel("Percentage of answers")
plt.legend()
plt.show()
```



In [44]:
```
print("Maximum length of answers : ", max(vqa_train_df['num_words_in_ans']))
print("Minumum length of answers : ", min(vqa_train_df['num_words_in_ans']))
print("Mean length of answers : ", np.mean(vqa_train_df['num_words_in_ans']))
```

```
Maximum length of answers :  18
Minumum length of answers :  1
Mean length of answers :  1.0996175834972743
```

In [45]:
```
# Boxplot
plt.figure(figsize=(12,8))
sns.boxplot(y=vqa_train_df['num_words_in_ans'], color='g')
plt.title("Boxplot of Answers Lengths")
plt.ylabel("Number of words in answers")
plt.show()
```



NOTE :

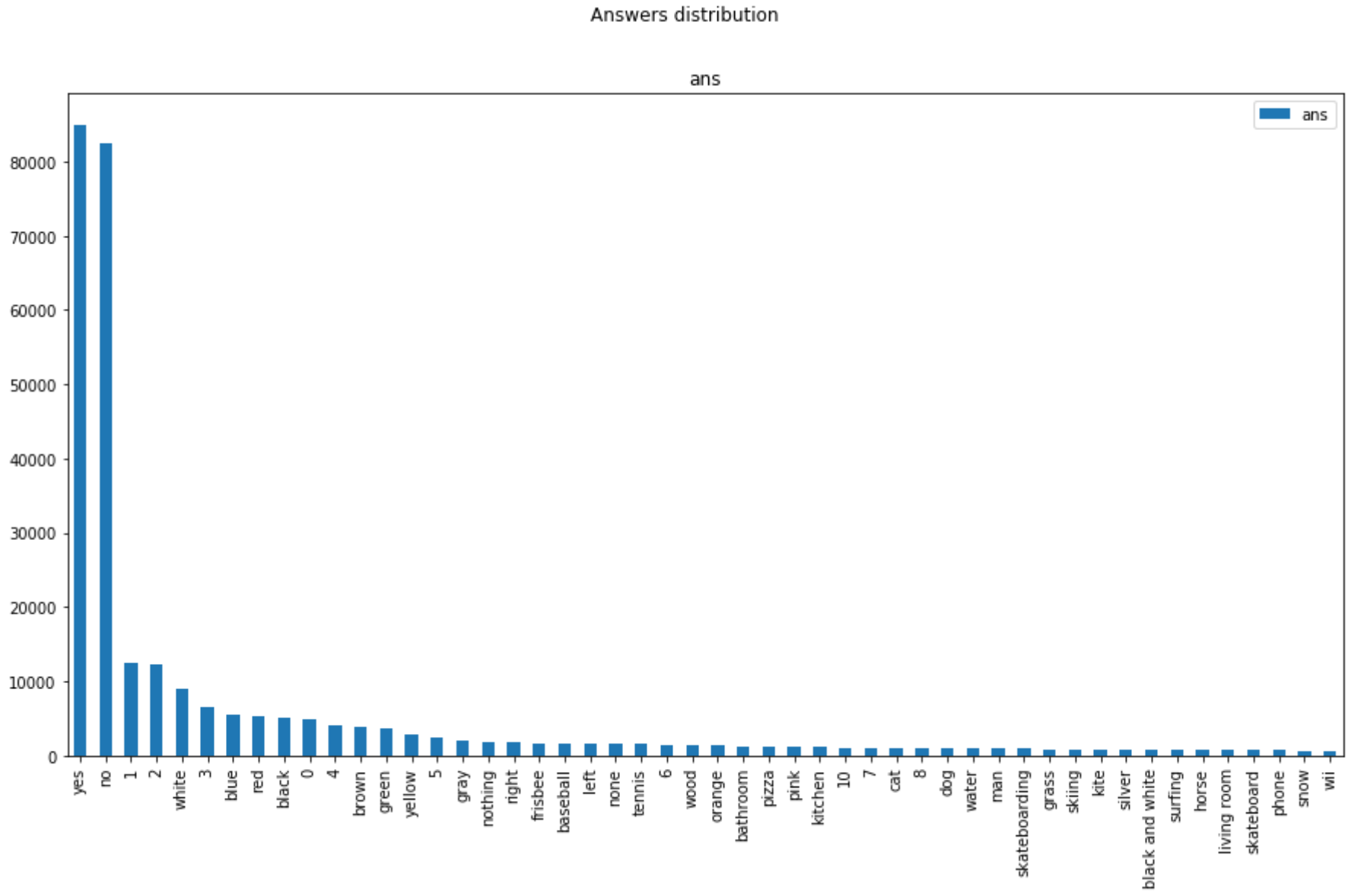
1. As we can see from the PDF that most of the answers are of length 1 or 2 words
2. There are very few questions with long answers.
3. Most answers range from 1 to 3 words.

Q. If we want to make this a multiclass classification task then how many classes do we need?

3.12 Let's see the distribution of the top 50 answers

In [46]:
```
fig, ax = plt.subplots(figsize=(15,8))
pd.DataFrame(vqa_train_df['ans'].value_counts()[0:50,]).plot(kind='bar',
                                                             ax=ax,
                                                             subplots=True,
                                                             title='Answers distribution')
```

Out[46]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f0b52d74470>],
      dtype=object)
```



3.12 Counting the top 1000 answers in the given dataset

In [47]:
```
vqa_answers_df = pd.DataFrame(vqa_train_df['ans'].value_counts()[:1000,]).reset_index()
vqa_answers_df.rename(columns={'index':'answers','ans':'count'}, inplace=True)
vqa_answers_df['percentage_count'] = pd.DataFrame(vqa_train_df['ans'].value_counts(normalize=True)[:1000,]).reset_inc
```

In [48]:
```
vqa_answers_df
```

Out[48]:

	answers	count	percentage_count
0	yes	84978	0.191497
1	no	82516	0.185949
2	1	12540	0.028259
3	2	12215	0.027526
4	white	8916	0.020092
5	3	6536	0.014729
6	blue	5455	0.012293
7	red	5201	0.011720
8	black	5066	0.011416
9	0	4977	0.011216
10	4	4118	0.009280
11	brown	3814	0.008595
12	green	3750	0.008451
13	yellow	2792	0.006292
14	5	2367	0.005334
15	gray	2113	0.004762
16	nothing	1814	0.004088
17	right	1766	0.003980
18	frisbee	1641	0.003698
19	baseball	1597	0.003599
20	left	1565	0.003527
21	none	1563	0.003522
22	tennis	1502	0.003385
23	6	1455	0.003279
24	wood	1449	0.003265
25	orange	1425	0.003211
26	bathroom	1230	0.002772
27	pizza	1203	0.002711
28	pink	1202	0.002709
29	kitchen	1093	0.002463
...	...	...	...
970	white and gray	28	0.000063
971	mutt	28	0.000063
972	oil	28	0.000063
973	railing	28	0.000063
974	children	28	0.000063
975	regular	28	0.000063
976	reins	28	0.000063
977	fur	28	0.000063
978	cones	28	0.000063
979	tattoo	28	0.000063
980	cross country	28	0.000063
981	tired	28	0.000063
982	foreground	28	0.000063
983	garage	28	0.000063
984	log	28	0.000063
985	snowy	28	0.000063
986	plaster	28	0.000063
987	bush	28	0.000063
988	boxing	27	0.000061
989	corner	27	0.000061
990	peace	27	0.000061
991	drywall	27	0.000061
992	dugout	27	0.000061
993	on grass	27	0.000061
994	army	27	0.000061
995	dishes	27	0.000061

	answers	count	percentage_count
996	swinging	27	0.000061
997	town	27	0.000061
998	on pole	27	0.000061
999	roman numerals	27	0.000061

1000 rows × 3 columns

In [49]:
```
# Sum of all the top 1000 answer counts
vqa_answers_df['count'].sum()
```

Out[49]: 388158

In [50]:
```
print("Percentage of questions convered by top 1000 answers : ",
      (vqa_answers_df['count'].sum()/len(vqa_train_df))*100, "%")
```

```
Percentage of questions convered by top 1000 answers :  87.47084553032403 %
```

NOTE :

1. As we can see we can do a 1000 class classification task as many of the questions (around 388158 out of 443757) will be covered by this classification task.
2. Most of the answers in the dataset is YES/NO answers and some numerical values. Others are just one word answers and their count is very few.

CONCLUSION:

1. Open ended questions result in a diverse set of possible answers.
2. For some questions a simple yes or no is sufficient but for many questions (like question starting with "WHAT") the answer diversity increases.
3. Some questions are answered by a short phrase as well but such answers are very less as most of the answers are just 1 word.
4. As per the paper :
   - For every question there were 10 answers collected and finally they were evaluated using the following accuracy metric :
   - accuracy = min{(Number of humans taht provided that answers / 3) , 1}
5. There are many repeated questions for the images (as shown in the EDA) because the question may be same but answer varies as per the given image.
6. There are 65 `questions type` and 3 `answer_types` in the dataset.
7. Question length for most of the questions is between 5-8 and Answer length for most of the questions is just 1 or 2 words.

In [ ]:

In [ ]:

In [ ]: