Upvotes Survey Code:

Crude code for majority vote, since weighted votes cannot be done (since we don’t have qual_ctrl questions)

‘csv’ is the data from Mechanical Turk (as the title of a csv file)

‘hit_number’ is the number of HITs per assignment

```
def majority_vote(data, hit_number):
 data = data.sort_values(by=['Destination'], ignore_index = True)
 # results will only include locations that were upvoted sufficiently
 results = []
 for hit in range(len(data['Do you recommend visiting?'])):
   yes_count = 0
   for idx in range(hit_number):
     if (data['Do you recommend visiting?'][hit + idx] == 'Yes'):
       if (data['Which best represents your relation to the
destination'][hit + idx]== 'I have been there'):
         yes_count = yes_count + 2
       else:
         yes_count = yes_count + 1
   if (yes_count > hit_number):
     results.append((data['Location'][x]))
 return results

def main():
 data = pd.read_csv('upvoting.csv')
 updated_destinations = majority_vote(data, 5)
 updated_df = pd.DataFrame(updated_destinations, columns =
['Destination'])
 updated_df.to_csv('new_destinations.csv')

if __name__ == '__main__':
   main()

 # With the new csv, we have a new (smaller) dataset for either further
upvoting or refining
```

Code for Quality Control / Refining Recommendations:

Criteria: Similar to third quality control algorithm from homework 7.

A location is classified as ‘good’ if: Google_maps = TRUE and Visit_likelihood >= 50

A photograph is classified as ‘good’ if: Photo_relevancy = Yes and Photo_rating >= 4

```
def select_qualified_locations(mturk_res):
 # remove all unnecessary columns
 df = mturk_res[['WorkerId', 'Location', 'Google_maps', 'Turker_visited',
                                          'Photo_relevancy',
'Photo_rating', 'Visit_likelihood']]
 # make a new column of whether or not location is good
 df["Good_location"] = df.apply(lambda x : True
               if (x['Google_maps'] == True and x['Visit_likelihood'] >=
50)
               else False, axis = 1)
 # make a new column of whether or not photo is good
 df["Good_photo"] = df.apply(lambda x : True
               if (x['Photo_Relevancy'] == "Yes" and x['Photo_rating'] >=
4)
               else False, axis = 1)
 # drop rows if location is not good
 df.drop(df[df.Good_location != True].index, inplace=True)
 # drop rows if photo is not good
 df.drop(df[df.Photo_relevancy == "No"].index, inplace=True)
 df.drop(df[df.(Photo_relevancy == "Yes" and Photo_ratin <= 3)].index,
inplace=True)
 return df
```