Testing of methods contained within the pymongo/mongodb docs:

```
import os
from datetime import datetime
import json
from bson import json_util
from bson.json_util import dumps
import pprint
from flask import Flask, redirect, render_template, request, flash, url_for
from flask_pymongo import PyMongo
from bson.objectid import ObjectId
import pymongo
import datetime
from mongo_datatables import DataTables
from pymongo import MongoClient

app = Flask(__name__)
app.config["MONGO_DBNAME"] = 'recipe_book'
app.config["MONGO_URI"] =
'mongodb://recipes:18Recipes18@ds115592.mlab.com:15592/recipe_book'

mongo = PyMongo(app)
```

#count documents

```
def count_data():
   #count all the recipes in the db
   totalCount = mongo.db.recipes.count()
   print(totalCount)
   #count the first 5
   totalLimitCount = mongo.db.recipes.find().limit(5).count(True)
   print(totalLimitCount)
count_data()
```

code above returns 16 and 5 respectively.

#find a recipe based on upvote criteria

```
def find_recipes():
        recipes=mongo.db.recipes.find({"upvotes": 2})
        for recipe in recipes:
                pprint.pprint(recipe)
find_recipes()
```

This prints out all attributes all recipes (with an upvote value of 2) and their attributes. (In this case 1 recipe was found).

# counts vegetarian dishes

```
def find_data():
        recipes=mongo.db.recipes.find({"Suitable_for_Vegetarians": { '$in': [ "Yes",
“yes" ] } } ).count()
        print(recipes)
find_data()
```

This prints out the number of vegetarian recipes. (In this case 11).

#finds all recipes from nominated countries

```
def find_data():
 recipes = mongo.db.recipes.find( { "Country_of_origin": { "$in": [ "India", "Germany" ] } } )
          igCount = 0
       for recipe in recipes:
           igCount +=1
          pprint.pprint(recipe)
```
This prints out 3 number recipes. (In this case 2 from Germany and 1 from India).
find_data()

#retrieve all documents in the collection where the country of origin is america and the total time is less than 35 minutes
```
def find_data():
          recipes = mongo.db.recipes.find( { "Country_of_origin": "America", "Total_time": { "$lt":
          "35" } } )
          for recipe in recipes:
                    pprint.pprint(recipe)
find_data()
```
prints out all attributes of the recipes where country of origin is america and total time is less than 30 minutes.
#find recipes from america where either the name starts with a b or the total time is less than 35 mins.
```
def find_data():
   recipes = mongo.db.recipes.find(
        {
    "Country_of_origin": "America",
    "$or": [ { "Total_time": { "$lt": "35" } }, { "Recipe_name": "/^b/" } ]
} )
   for recipe in recipes:
      pprint.pprint(recipe)
find_data()
```
prints out all attributes for the found recipes (in this case 4 were found to either begin with b or take less than 35 minutes to cook and prepare.

#find all recipes added after 21/08/18
```
def find_data():
recipes = mongo.db.recipes.find({"Date_added": {"$gte": 'Tuesday, 21, August, 2018'}})
   for recipe in recipes:
      count +=1
      pprint.pprint(recipe)
      print(count)
find_data()
```
the above function found 3 recipes (in this instance)

# sort and count by country of origin
```
def sort_data():
   count = 0
   Vegcount = 0
   #code below works, finds, sorts and counts
   recipes=mongo.db.recipes.find().sort("Country_of_origin", pymongo.DESCENDING)
   for recipe in recipes:
      count  +=1
      pprint.pprint(recipe)
   print (count)
```

Sorts all recipes by country of origin (in descending order) and outputs the total number (in this case 16)

```
sort_data()
```

#code below also sorts descending

```
recipes=mongo.db.recipes.find()
   for recipe in recipes.sort("Country_of_origin", pymongo.DESCENDING):
      pprint.pprint(recipe)
sort_data()
```

#print out specified attributes relating to suitable for vegans

```
def retrieve_attributes():
   recipes=mongo.db.recipes.find( { "Suitable_for_Vegans": {"$in": ["Yes", "yes"]}},
{ "Recipe_name": 1, "Country_of_origin": 1, "_id": 0 } )
   for recipe in recipes:
      pprint.pprint(recipe)
   print(type(recipe))
retrieve_attributes()
```

code above prints out the two specified attributes for all found recipes relating to vegans (in this case 6 records found) Data type is a dictionary.

```
         {u'Country_of_origin': u'China',
         u'Recipe_name': u'Homemade Chinese Curry Sauce'}
         {u'Country_of_origin': u'India',
         u'Recipe_name': u'Homemade Indian Curry Sauce'}
         {u'Country_of_origin': u'America', u'Recipe_name': u'Potato Salad'}
         {u'Country_of_origin': u'Germany',
         u'Recipe_name': u'Spinach Soup with Roasted Tofu'}
         {u'Country_of_origin': u'America', u'Recipe_name': u'Black Beans with Rice'}
         {u'Country_of_origin': u'Germany',
         u'Recipe_name': u'Flower Sprouts with Baked Potatoes and Sauce with Dried Tomatoes'}
         <type 'dict'>
```

#code below prints out the same criteria as above in string format

```
   recipes=mongo.db.recipes.find( { "Suitable_for_Vegans": {"$in": ["Yes", "yes"]}},
{ "Recipe_name": 1, "Country_of_origin": 1, "_id": 0 } )
   data= recipes
   json_data = []
   for datum in data:
      json_data.append(datum)
   json_data = json.dumps(json_data, default=json_util.default)
   # return json_data
   pprint.pprint(json_data)
   print(type(json_data))
```

```
'[{"Country_of_origin": "China", "Recipe_name": "Homemade Chinese Curry Sauce"},
{"Country_of_origin": "India", "Recipe_name": "Homemade Indian Curry Sauce"},
{"Country_of_origin": "America", "Recipe_name": "Potato Salad"}, {"Country_of_origin":
"Germany", "Recipe_name": "Spinach Soup with Roasted Tofu"}, {"Country_of_origin":
"America", "Recipe_name": "Black Beans with Rice"}, {"Country_of_origin": "Germany",
"Recipe_name": "Flower Sprouts with Baked Potatoes and Sauce with Dried Tomatoes"}]'
<type 'str'>
```

#group the allergens, count them and sort descending

```
def aggregate_data():
    pipeline = [
        {"$group": {"_id": "$Allergens", "count": {"$sum": 1}}},
        {"$sort": SON([("count", -1), ("_id", -1)])}
        ]
    pprint.pprint(list(mongo.db.recipes.aggregate(pipeline)))
aggregate_data()
```

Output:

```
[{u'_id': u'None Known', u'count': 7},
 {u'_id': u'Contains Gluten', u'count': 3},
 {u'_id': u'Contains Egg, Gluten, Nuts', u'count': 2},
 {u'_id': u'Contains Sesame, Soy\r\n', u'count': 1},
 {u'_id': u'Contains Lupin, Mustard', u'count': 1},
 {u'_id': u'Contains Fish, Celery, Clams', u'count': 1},
 {u'_id': u'Contains Celery', u'count': 1}]
```

# return all recipes in string format showing only attributes listed in the projection code below works.

```
def recipe_book():
    recipes=mongo.db.recipes.find({}, { "category_name": 1, "Total_time": 1, "Date_added": 1,
"Allergens": 1, "Suitable_for_Vegans": 1, "Suitable_for_Vegetarians": 1, "Recipe_name": 1,
"Country_of_origin": 1, "_id": 0 })
    #output as a string list  for use in dc js charting
    json_recipes = []
    for attribute in recipes:
        json_recipes.append(attribute)
        #convert object to string
    json_recipes = json.dumps(json_recipes, default=json_util.default)
    #return json_recipes
    pprint.pprint(json_recipes)
    print(type(json_recipes))
recipe_book()
```