Branch: **master** Find file Copy path

buckets / **bucket_collection_test.py**

trodicaro Edits for running the original data provided instead of my testing sa…

99770dc on Mar 9, 2018

1 contributor

Raw Blame History

135 lines (110 sloc) 6.44 KB

```python
import unittest
import os.path
import json
import random
from bucket_collection import Bucket
from bucket_collection import BucketCollection

# class BucketTest(unittest.TestCase):
#     def test_init_creates_bucket_object(self):
#         pass
#     def test_init_assigns_key(self):
#         pass

class BucketCollectionTest(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.results_file = "results.json"
        # cls.results_file = "min_results.json"
        cls.results_filepath = os.path.join(os.path.dirname(__file__), cls.results_file)

        bucket_collection = BucketCollection("purchase_buckets.csv", "purchase_data.csv")
        # bucket_collection = BucketCollection("min_buckets.csv", "min_purchases.csv")
        bucket_collection.to_file(cls.results_file)

        with open(cls.results_file) as file:
            results_json = json.loads(file.read())

        cls.data_dictionary = {}
        for item in results_json:
            if item['bucket'] in cls.data_dictionary:
                cls.data_dictionary[item['bucket'] + '-dup'] = item['purchases']
            else:
                cls.data_dictionary[item['bucket']] = item['purchases']
    # @classmethod
    # def tearDownClass(cls):
    #     print("Calling tearDown")
    #     os.remove(cls.results_file)

    def test_results_file_creation(self):
        "Tests that a result file is generated"
        self.assertTrue(os.path.isfile(self.results_file))

    def test_results_json_has_content(self):
        # alternate implementation: when loading JSON from file, instead of fail, show error
        # bucket_collection = BucketCollection("purchase_buckets.csv", "purchase_data.csv")
        # https://codeblogmoney.com/validate-json-using-python/
        # https://stackoverflow.com/questions/23344948/python-validate-and-format-json-files
        self.assertTrue(len(self.data_dictionary) > 0)

    def test_generic_bucket_existence(self):
        "Test that generic bucket was created"
        self.assertIn("*,*,*", self.data_dictionary.keys())

    def test_buckets_generate_in_desired_order(self):
        # check *,*,* is first key
        keys_iterator = iter(self.data_dictionary.keys())
        self.assertTrue(next(keys_iterator) == "*,*,*")
        # check first bucket in file provided is next key
        self.assertTrue(next(keys_iterator) == "McGraw-Hill,5,40_day")
        pass

    def test_a_purchase_is_found_only_once(self):
        test_purchases = [
            '98765,0862728122370,OPENSTAX,CLT,5,150_day,2017-05-31 14:21:29.560404',
            '98771,1899596499745,PEARSON,MIA,3,110_day,2017-05-23 09:16:43.560846',
            '99377,8660464769977,PEARSON,JFK,2,40_day,2017-02-10 15:04:03.578055',
            '98795,9277080469051,MCGRAW-HILL,MSP,6,120_day,2017-04-02 11:05:31.561470',
            '98775,7192583653601,SCIPUB,BOS,8,140_day,2017-08-03 14:02:28.560950',
            '98835,6544295182149,MACMILLAN,CLE,4,40_day,2017-01-10 14:08:55.562501',
            '99680,8193774926972,PEARSON,SNA,7,10_day,2017-07-10 07:07:11.587228',
            '98819,9793386372887,PENGUIN RANDOMHOUSE,DTW,3,90_day,2017-07-14 14:06:01.562089',
            '99999,9999999999999,MACMILLAN,MIA,3,110_day,2017-05-23 09:16:43.560846',
            '98815,8022139588957,ENGLISH PUBLICATIONS,DTW,10,120_day,2017-08-09 12:42:30.561986',
            '98793,3455843886681,ENGLISH PUBLICATIONS,MCO,4,60_day,2017-05-16 08:51:17.561418',
            '99191,7848537371773,PENGUIN RANDOMHOUSE,MIA,4,30_day,2017-05-21 10:01:19.571428']
        test_purchase = random.choice(test_purchases)
        all_purchases = self.data_dictionary.values()
        self.assertFalse(test_purchase in all_purchases)

    def test_purchases_ordered_by_order_id(self):
        random_bucket_key, bucket_purchases = random.choice(list(self.data_dictionary.items()))
        ids = list(map(lambda purchase: int(purchase.split(',')[0]), bucket_purchases))
        self.assertTrue(sorted(ids) == ids)

    def test_publisher_price_duration_bucket(self):
        "Most specific"
        test_string = "99145,0926889346680,MCGRAW-HILL,PHX,6,30_day,2017-08-31 12:52:41.570232"
        self.assertIn(test_string, self.data_dictionary["McGraw-Hill,6,30_day"])

    def test_publisher_duration_bucket(self):
        test_string = "98835,6544295182149,MACMILLAN,CLE,4,40_day,2017-01-10 14:08:55.562501"
        self.assertIn(test_string, self.data_dictionary["Macmillan,*,40_day"])

    def test_publisher_price_bucket(self):
        test_string = "98795,9277080469051,MCGRAW-HILL,MSP,6,120_day,2017-04-02 11:05:31.561470"
        self.assertIn(test_string, self.data_dictionary["McGraw-Hill,6,*"])

    def test_price_duration_bucket(self):
        test_string = "98819,9793386372887,PENGUIN RANDOMHOUSE,DTW,3,90_day,2017-07-14 14:06:01.562089"
        self.assertIn(test_string, self.data_dictionary["*,3,90_day"])

    def test_publisher_only_bucket(self):
        test_string = "98771,1899596499745,PEARSON,MIA,3,110_day,2017-05-23 09:16:43.560846"
        self.assertIn(test_string, self.data_dictionary["Pearson,*,*"])

    def test_publisher_only_bucket_with_upper_lower_letters(self):
        test_string = "98775,7192583653601,SCIPUB,BOS,8,140_day,2017-08-03 14:02:28.560950"
        self.assertIn(test_string, self.data_dictionary["SciPub,*,*"])

    def test_publisher_only_bucket_with_dash(self):
        test_string = "98795,9277080469051,MCGRAW-HILL,MSP,6,120_day,2017-04-02 11:05:31.561470"
        self.assertIn(test_string, self.data_dictionary["McGraw-Hill,6,*"])

    def test_publisher_only_bucket_with_space(self):
        test_string = "99191,7848537371773,PENGUIN RANDOMHOUSE,MIA,4,30_day,2017-05-21 10:01:19.571428"
        self.assertIn(test_string, self.data_dictionary["Penguin Randomhouse,*,30_day"])

    # def test_duration_only_bucket(self):
    #     test_string = "99999,9999999999999,MACMILLAN,MIA,3,110_day,2017-05-23 09:16:43.560846"
    #     self.assertIn(test_string, self.data_dictionary["*,*,110_day"])

    def test_price_only_bucket(self):
        test_string = "98815,8022139588957,ENGLISH PUBLICATIONS,DTW,10,120_day,2017-08-09 12:42:30.561986"
        self.assertIn(test_string, self.data_dictionary["*,10,*"])

    def test_catch_all_bucket(self):
        test_string = "98765,0862728122370,OPENSTAX,CLT,5,150_day,2017-05-31 14:21:29.560404"
        self.assertIn(test_string, self.data_dictionary["*,*,*"])

    def test_edge_case_repeated_bucket(self):
        self.assertFalse(self.data_dictionary["SciPub,*,*-dup"])

if __name__ == "__main__":
    unittest.main()
```