Presto Engineering Challenge

Note: Although the question uses a Pythonic Tone, you can use any programming language of your choice to implement the solution

Introduction

Joe's Grill is a very important customer to Presto. They want us to manage their website. One of the most important pages on their website is the “Menu” page.The menu gets updated every night at the restaurant and Joe’s grill wants the updated menu published to their website every morning. Joe’s grill will upload their menu in a JSON formatted file (Let’s call it menu.json) to our ftp server every night.

Presto wants the engineers to develop a program to take this menu.json file and store the data in a relational database. The data model for the database tables is defined as follows.

Data model

```
        Restaurant : id( pkey ), name

        items: id(pkey), name
        categories: id(pkey), name

        category_items:
            id(pkey), category_id (fkey categories), item_id(fkey items), rest_id, price
```

menu.json

```
 "restaurant":"Joe's Grill" {

{
  "Category”:
[
         {
         “name”: “Appetizer”,
         },
         {
         “name”: “Entree”
         }
  ]
   ],
   "item”:[
      {
         "name":"French Fries",
         "category":"Appetizer"
      },
      {
         "name":"Onion Rings",
         "category":"Appetizer"
      },
      {
         "name":"Sandwich",
         "category":"Entree"
      },
      {
         "name":"Tacos",
         "category":"Entree"
      },
      {
         "name":"Ice Cream Sundae",
         "category":"Dessert"
      }
      ],
   }
}
```

Existing classes:

```
class Category:
      def __init__(self, name):
                  self.name = name

class Item:
          def __init__(self, name, category_name):
                  self.name = name
                  self.category_name = category_name
```

Helper Functions that Already exist

// json.get_categories(file_name) -> returns array of active category objects (category.name)

```
// json.get_items(file_name) --> returns array of item object (item.name, item.category_name)

// db.write(table_name, <a data dict or an array of data dicts>)

//db.write('categories', {'name': 'c1'})
```

Note:The keys of the dictionary are the table column names, the values are the row values in the database table

```
Class dbWrite()
    Def __init__(  self, jsonFile ):
        self.jsonFile = jsonFile
        Self.output = parse( jsonFile )
        Self.db = None

    Def write() :

        ‘’’
        [
             {
             “name”: “Appetizer”,
             },
             {
             “name”: “Entree”
             }
         ]      ‘’’

        Count = 0

        Categories = [ Category( c.get( “name” ) ) for c in self.output.get( “Category” ) ]
        categoryCountMap = {}
        For category in categories:
            categoryCountMap[ Category(category) ] = count
            db.write( “categories”, [ count, category ] )
            Count += 1

        Count = 0
        itemCategoryMap = {}
        itemCountMap = {}

        ItemsFromJson = self.output.get( “Item” )
```

```
For data in ItemsFromJson:
        item = Item( data.get( “name” ),
                     data.get( “category” )  )
        category = Category( data.get( “category” ) )

        itemCategoryMap[ item ] = category

        itemCountMap[  Items( item ) ] = count

        db.write( “items”, [ count, item ] )

        If categoryCountMap.get( category ) != None:
                db.write( “category_items”, [ count,
                                              itemCountMap.get( item ),
                                              categoryCountMap.get( category ) ]
```