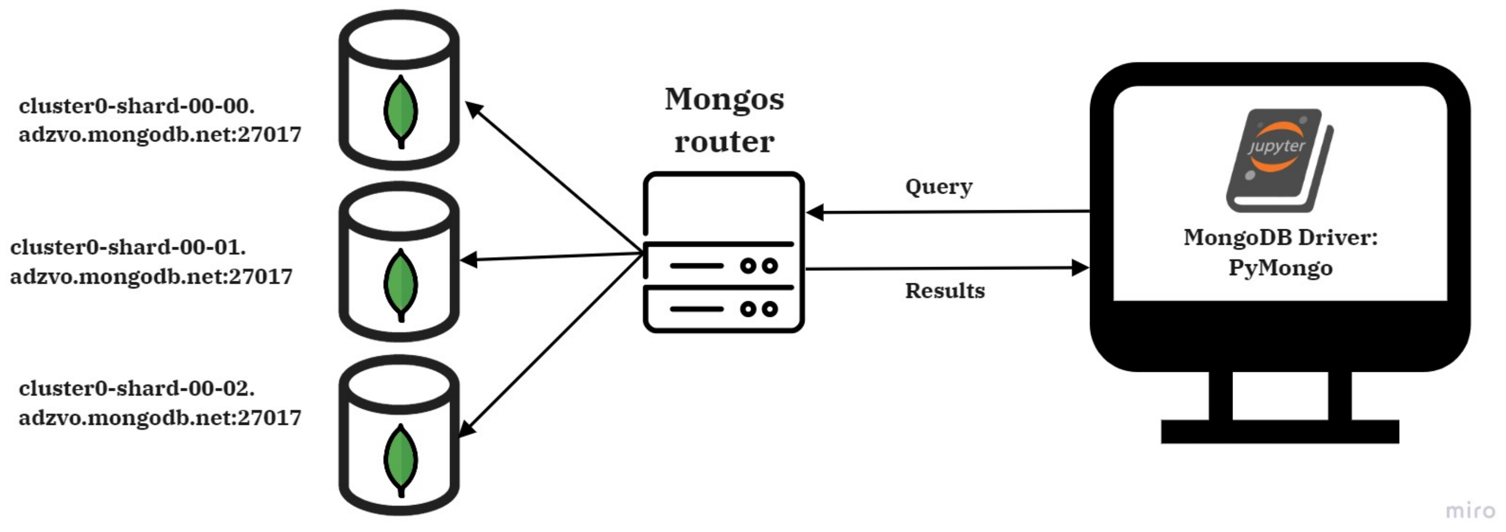
TheMealDB

Table of Contents

1. Introduction

The meal database https://www.themealdb.com/ (https://www.themealdb.com/) is an open, crowd-sourced database of Recipes from around the world that offers a free JSON API. The raw JSON data fetched from the API is not ideal for querying. In this report, we modify document structure into a more query-friendly one. In addition, we conduct an analysis where we answer some questions about cuisines, ingredients and meal categories!

The system architecture is shown in the figure below. We connect to the Mongos router using PyMongo as a driver, which in turn routes the query to the respective shards and returns back the query results. Our MongoDB cluster is sharded into three shards as depicted in the figure.



We start by importing the required modules

```python
# first, some imports and settings ...

import pymongo
import string
import requests
import time
import pandas as pd
from pprint import pprint
import warnings
warnings.filterwarnings('ignore')
pd.set_option('max_colwidth', 250)
```

Then we make the connection to the Mongos router.

```
# connecting to mongodb
connection_string = "mongodb+srv://yasmine:hslu5678@cluster0.adzvo.mongodb.net/?retryWrites=true&w=majority"
client = pymongo.MongoClient(connection_string)
db = client['the_meal_db']
```

```
# quick look

db.list_collection_names()
```

```
['meals_raw']
```

```
db.meals_raw.count_documents({})
```

```
0
```

2. Extract. Load. Transofrm.

2.1. Extract from API and Load into MongoDB cluster

The meal database does not offer an API call that can fetch all available meals at once. Therefore, we have to iterate through the list of meals alphabetically.

```
# creating a list of the alphabet letters to use it iteratively in the API's URL

list_characters = [char for char in string.ascii_lowercase]
alphabet = list_characters
```

```
# collecting all available meals from "the meal db" API by inserting one alphabet letter at a time in the URL string
# populating the 'meals_raw' collection in 'the meal db' database

collection = db.meals_raw

for letter in alphabet:
    url = f'https://www.themealdb.com/api/json/v1/1/search.php?f={letter}'
    r = requests.get(url)
    data = r.json()
    if data['meals'] is not None:
        collection.insert_many(data['meals'])
    time.sleep(2)
```

```
# checking how many documents were inserted

db.meals_raw.count_documents({})
```

```
282
```

By taking a look at one entry...

```
# taking a look at one entry

pprint(collection.find_one({}))
```

```
{'_id': ObjectId('628a4dabe49447e6a6e79a5a'),
 'dateModified': None,
 'idMeal': '52893',
 'strArea': 'British',
 'strCategory': 'Dessert',
 'strCreativeCommonsConfirmed': None,
 'strDrinkAlternate': None,
 'strImageSource': None,
 'strIngredient1': 'Plain Flour',
 'strIngredient10': '',
 'strIngredient11': '',
 'strIngredient12': '',
 'strIngredient13': '',
 'strIngredient14': '',
 'strIngredient15': '',
 'strIngredient16': '',
 'strIngredient17': '',
 'strIngredient18': '',
 'strIngredient19': '',
 'strIngredient2': 'Caster Sugar',
 'strIngredient20': '',
 'strIngredient3': 'Butter',
 'strIngredient4': 'Braeburn Apples',
 'strIngredient5': 'Butter',
 'strIngredient6': 'Demerara Sugar',
 'strIngredient7': 'Blackberrys',
 'strIngredient8': 'Cinnamon',
 'strIngredient9': 'Ice Cream',
 'strInstructions': 'Heat oven to 190C/170C fan/gas 5. Tip the flour and sugar '
                    'into a large bowl. Add the butter, then rub into the '
                    'flour using your fingertips to make a light breadcrumb '
                    'texture. Do not overwork it or the crumble will become '
                    'heavy. Sprinkle the mixture evenly over a baking sheet '
                    'and bake for 15 mins or until lightly coloured.\r\n'
                    'Meanwhile, for the compote, peel, core and cut the apples '
                    'into 2cm dice. Put the butter and sugar in a medium '
                    'saucepan and melt together over a medium heat. Cook for 3 '
                    'mins until the mixture turns to a light caramel. Stir in '
                    'the apples and cook for 3 mins. Add the blackberries and '
                    'cinnamon, and cook for 3 mins more. Cover, remove from '
                    'the heat, then leave for 2-3 mins to continue cooking in '
                    'the warmth of the pan.\r\n'
                    'To serve, spoon the warm fruit into an ovenproof gratin '
                    'dish, top with the crumble mix, then reheat in the oven '
                    'for 5-10 mins. Serve with vanilla ice cream.',
 'strMeal': 'Apple & Blackberry Crumble',
 'strMealThumb': 'https://www.themealdb.com/images/media/meals/xvsurr1511719182.jpg',
 'strMeasure1': '120g',
 'strMeasure10': '',
 'strMeasure11': '',
 'strMeasure12': '',
 'strMeasure13': '',
 'strMeasure14': '',
 'strMeasure15': '',
 'strMeasure16': '',
 'strMeasure17': '',
 'strMeasure18': '',
 'strMeasure19': '',
 'strMeasure2': '60g',
 'strMeasure20': '',
 'strMeasure3': '60g',
 'strMeasure4': '300g',
 'strMeasure5': '30g',
 'strMeasure6': '30g',
 'strMeasure7': '120g',
 'strMeasure8': '¼ teaspoon',
 'strMeasure9': 'to serve',
 'strSource': 'https://www.bbcgoodfood.com/recipes/778642/apple-and-blackberry-crumble',
 'strTags': 'Pudding',
 'strYoutube': 'https://www.youtube.com/watch?v=4vhcOwVBDO4'}
```

.... we see that the structure is far from desirable and needs some processesing before any meaningful analysis can be conducted

2.2. Transformation

The trasnformation steps are as follows:

1. Collect all ingredients into a single array
2. Collect all measures into a single array
3. Remove empty strings and null values
4. Zip both arrays into a new field 'recipe', so that we have the format {ingredient:measure}
5. Rename field names
6. Remove unnecessary fields

```python
#creating a list of Ingredient field names to avoid adding them one by one in the aggregation pipelines

numbers = [i for i in range(1,21)]

ingredient_list = []

for number in numbers:
    x = str('$strIngredient'+str(number))
    if x is not None:
        ingredient_list.append(x)
ingredient_list
```

```
['$strIngredient1',
 '$strIngredient2',
 '$strIngredient3',
 '$strIngredient4',
 '$strIngredient5',
 '$strIngredient6',
 '$strIngredient7',
 '$strIngredient8',
 '$strIngredient9',
 '$strIngredient10',
 '$strIngredient11',
 '$strIngredient12',
 '$strIngredient13',
 '$strIngredient14',
 '$strIngredient15',
 '$strIngredient16',
 '$strIngredient17',
 '$strIngredient18',
 '$strIngredient19',
 '$strIngredient20']
```

```python
# then doing the same for measures

measure_list = []

for number in numbers:
    x = str('$strMeasure'+str(number))
    if x is not None:
        measure_list.append(x)
```

```python
# The same but this time without the preceeding dollar sign '$'

ingredient_list_no_dollar = []

for number in numbers:
    x = str('strIngredient'+str(number))
    if x is not None:
        ingredient_list_no_dollar.append(x)


measure_list_no_dollar = []

for number in numbers:
    x = str('strMeasure'+str(number))
    if x is not None:
        measure_list_no_dollar.append(x)
```

First, we gather all ingredients and all measures in arrays, and remove the individual 'strIngredient' and 'strMeasure' fields. We push the output into a a new collection "meals".

```python
# adding ingredients and measure arrays
# removing individual fields

db.meals_raw.aggregate([
    {"$addFields":
     {"ingredients" : [i for i in ingredient_list],
      "measures" : [i for i in measure_list]},
    },
    {"$project": {field: 0 for field in ingredient_list_no_dollar}},
    {"$project": {field: 0 for field in measure_list_no_dollar}},
    {"$out": "meals"}])
```

```
<pymongo.command_cursor.CommandCursor at 0x1ce6f244c70>
```

```
pprint(db.meals.find_one({}))
```

```
{'_id': ObjectId('628a4dabe49447e6a6e79a5a'),
 'dateModified': None,
 'idMeal': '52893',
 'ingredients': ['Plain Flour',
                 'Caster Sugar',
                 'Butter',
                 'Braeburn Apples',
                 'Butter',
                 'Demerara Sugar',
                 'Blackberrys',
                 'Cinnamon',
                 'Ice Cream',
                 '',
                 '',
                 '',
                 '',
                 '',
                 '',
                 '',
                 '',
                 '',
                 '',
                 ''],
 'measures': ['120g',
              '60g',
              '60g',
              '300g',
              '30g',
              '30g',
              '120g',
              '¼ teaspoon',
              'to serve',
              '',
              '',
              '',
              '',
              '',
              '',
              '',
              '',
              '',
              '',
              ''],
 'strArea': 'British',
 'strCategory': 'Dessert',
 'strCreativeCommonsConfirmed': None,
 'strDrinkAlternate': None,
 'strImageSource': None,
 'strInstructions': 'Heat oven to 190C/170C fan/gas 5. Tip the flour and sugar '
                    'into a large bowl. Add the butter, then rub into the '
                    'flour using your fingertips to make a light breadcrumb '
                    'texture. Do not overwork it or the crumble will become '
                    'heavy. Sprinkle the mixture evenly over a baking sheet '
                    'and bake for 15 mins or until lightly coloured.\r\n'
                    'Meanwhile, for the compote, peel, core and cut the apples '
                    'into 2cm dice. Put the butter and sugar in a medium '
                    'saucepan and melt together over a medium heat. Cook for 3 '
                    'mins until the mixture turns to a light caramel. Stir in '
                    'the apples and cook for 3 mins. Add the blackberries and '
                    'cinnamon, and cook for 3 mins more. Cover, remove from '
                    'the heat, then leave for 2-3 mins to continue cooking in '
                    'the warmth of the pan.\r\n'
                    'To serve, spoon the warm fruit into an ovenproof gratin '
                    'dish, top with the crumble mix, then reheat in the oven '
                    'for 5-10 mins. Serve with vanilla ice cream.',
 'strMeal': 'Apple & Blackberry Crumble',
 'strMealThumb': 'https://www.themealdb.com/images/media/meals/xvsurr1511719182.jpg',
 'strSource': 'https://www.bbcgoodfood.com/recipes/778642/apple-and-blackberry-crumble',
 'strTags': 'Pudding',
 'strYoutube': 'https://www.youtube.com/watch?v=4vhcOwVBDO4'}
```

We see that ingredients and measures are gathered into arrays, however, there are empty strings and null values. We get rid of those by the pull operator.

```
db.meals.update_many({},
    {"$pull": {"ingredients":
               {"$in": ['', ' ', None]},
               "measures":
               {"$in": ['', ' ', None]}}
    }
     )
```

```
<pymongo.results.UpdateResult at 0x1ce6f2446d0>
```

```
pprint(db.meals.find_one({}))
```

```
{'_id': ObjectId('628a4dabe49447e6a6e79a5a'),
 'dateModified': None,
 'idMeal': '52893',
 'ingredients': ['Plain Flour',
                 'Caster Sugar',
                 'Butter',
                 'Braeburn Apples',
                 'Butter',
                 'Demerara Sugar',
                 'Blackberrys',
                 'Cinnamon',
                 'Ice Cream'],
 'measures': ['120g',
              '60g',
              '60g',
              '300g',
              '30g',
              '30g',
              '120g',
              '¼ teaspoon',
              'to serve'],
 'strArea': 'British',
 'strCategory': 'Dessert',
 'strCreativeCommonsConfirmed': None,
 'strDrinkAlternate': None,
 'strImageSource': None,
 'strInstructions': 'Heat oven to 190C/170C fan/gas 5. Tip the flour and sugar '
                    'into a large bowl. Add the butter, then rub into the '
                    'flour using your fingertips to make a light breadcrumb '
                    'texture. Do not overwork it or the crumble will become '
                    'heavy. Sprinkle the mixture evenly over a baking sheet '
                    'and bake for 15 mins or until lightly coloured.\r\n'
                    'Meanwhile, for the compote, peel, core and cut the apples '
                    'into 2cm dice. Put the butter and sugar in a medium '
                    'saucepan and melt together over a medium heat. Cook for 3 '
                    'mins until the mixture turns to a light caramel. Stir in '
                    'the apples and cook for 3 mins. Add the blackberries and '
                    'cinnamon, and cook for 3 mins more. Cover, remove from '
                    'the heat, then leave for 2-3 mins to continue cooking in '
                    'the warmth of the pan.\r\n'
                    'To serve, spoon the warm fruit into an ovenproof gratin '
                    'dish, top with the crumble mix, then reheat in the oven '
                    'for 5-10 mins. Serve with vanilla ice cream.',
 'strMeal': 'Apple & Blackberry Crumble',
 'strMealThumb': 'https://www.themealdb.com/images/media/meals/xvsurr1511719182.jpg',
 'strSource': 'https://www.bbcgoodfood.com/recipes/778642/apple-and-blackberry-crumble',
 'strTags': 'Pudding',
 'strYoutube': 'https://www.youtube.com/watch?v=4vhcOwVBDO4'}
```

Now we have the required arrays, without any empty strings or null values. It would have been desirable to transform the measures into integers, but the extreme heterogeneity of the documents with respect to documenting measures makes this an almost impossible task at least directly on MongoDB (obviously users could write freely without choosing from a dropdown menu).

Next we zip the ingredients and measures array together into a new field 'recipe' for easier access of information. This way we gathered information that was scattered across 40 fields into one field.

```
db.meals.aggregate([
    {"$addFields": {"recipe":
                    {"$arrayToObject":
                     {"$zip": {"inputs": ["$ingredients", "$measures"]}}
                    }
                   }
    },
    { "$out" : "meals"}
                    ])
```

```
<pymongo.command_cursor.CommandCursor at 0x1ce6f244310>
```

```
db.meals.find_one({}, {"recipe":1})
```

```
{'_id': ObjectId('628a4dabe49447e6a6e79a5a'),
 'recipe': {'Plain Flour': '120g',
  'Caster Sugar': '60g',
  'Butter': '30g',
  'Braeburn Apples': '300g',
  'Demerara Sugar': '30g',
  'Blackberrys': '120g',
  'Cinnamon': '¼ teaspoon',
  'Ice Cream': 'to serve'}}
```

Last step is to rename fields and drop those that are not needed.

```
#dropping unnecessary fields

db.meals.aggregate([
    {"$project":
     {"strDrinkAlternate":0, "strImageSource":0, "dateModified":0, "strCreativeCommonsConfirmed":0}
    },
    {'$out': "meals"}
])
```

```
<pymongo.command_cursor.CommandCursor at 0x1ce6f23b940>
```

```
# renaming fields

db.meals.update_many({}, {"$rename":{
                              "strMeal": "meal_name",
                              "strCategory": "category",
                              "strArea": "area",
                              "strInstructions": "instructions",
                              "strTags": "tags",
                              "strYoutube": "youtube_link",
                              "strSource": "source",
                              "strMealThumb": "image",
                              "idMeal": "meal_id"}
                         }
                    )
```

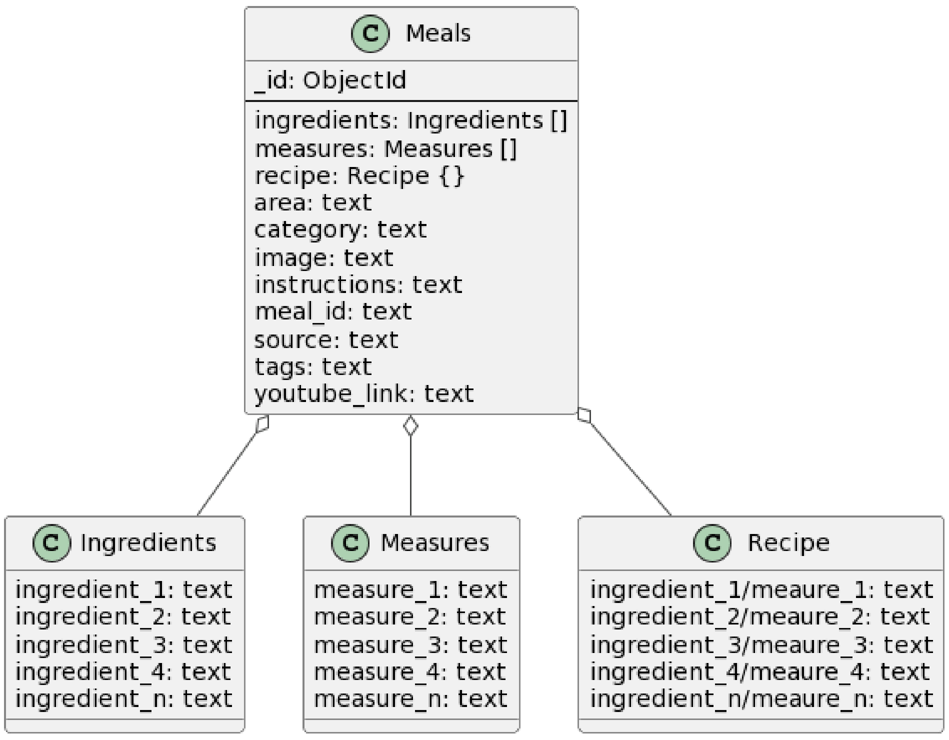
```
<pymongo.results.UpdateResult at 0x1ce6f117fa0>
```

```
#check the final structure

pprint(db.meals.find_one({}))
```

```
{'_id': ObjectId('628a4dabe49447e6a6e79a5a'),
 'area': 'British',
 'category': 'Dessert',
 'image': 'https://www.themealdb.com/images/media/meals/xvsurr1511719182.jpg',
 'ingredients': ['Plain Flour',
                 'Caster Sugar',
                 'Butter',
                 'Braeburn Apples',
                 'Butter',
                 'Demerara Sugar',
                 'Blackberrys',
                 'Cinnamon',
                 'Ice Cream'],
 'instructions': 'Heat oven to 190C/170C fan/gas 5. Tip the flour and sugar '
                 'into a large bowl. Add the butter, then rub into the flour '
                 'using your fingertips to make a light breadcrumb texture. Do '
                 'not overwork it or the crumble will become heavy. Sprinkle '
                 'the mixture evenly over a baking sheet and bake for 15 mins '
                 'or until lightly coloured.\r\n'
                 'Meanwhile, for the compote, peel, core and cut the apples '
                 'into 2cm dice. Put the butter and sugar in a medium saucepan '
                 'and melt together over a medium heat. Cook for 3 mins until '
                 'the mixture turns to a light caramel. Stir in the apples and '
                 'cook for 3 mins. Add the blackberries and cinnamon, and cook '
                 'for 3 mins more. Cover, remove from the heat, then leave for '
                 '2-3 mins to continue cooking in the warmth of the pan.\r\n'
                 'To serve, spoon the warm fruit into an ovenproof gratin '
                 'dish, top with the crumble mix, then reheat in the oven for '
                 '5-10 mins. Serve with vanilla ice cream.',
 'meal_id': '52893',
 'meal_name': 'Apple & Blackberry Crumble',
 'measures': ['120g',
              '60g',
              '60g',
              '300g',
              '30g',
              '30g',
              '120g',
              '¼ teaspoon',
              'to serve'],
 'recipe': {'Blackberrys': '120g',
            'Braeburn Apples': '300g',
            'Butter': '30g',
            'Caster Sugar': '60g',
            'Cinnamon': '¼ teaspoon',
            'Demerara Sugar': '30g',
            'Ice Cream': 'to serve',
            'Plain Flour': '120g'},
 'source': 'https://www.bbcgoodfood.com/recipes/778642/apple-and-blackberry-crumble',
 'tags': 'Pudding',
 'youtube_link': 'https://www.youtube.com/watch?v=4vhcOwVBDO4'}
```

The data structure can be represented by the class diagram below:


Meals
_id: ObjectId
ingredients: Ingredients []
measures: Measures []
recipe: Recipe {}
area: text
category: text
image: text
instructions: text
meal_id: text
source: text
tags: text
youtube_link: text
Ingredients
ingredient_1: text
ingredient_2: text
ingredient_3: text
ingredient_4: text
ingredient_n: text
Measures
measure_1: text
measure_2: text
measure_3: text
measure_4: text
measure_n: text
Recipe
ingredient_1/meaure_1: text
ingredient_2/meaure_2: text
ingredient_3/meaure_3: text
ingredient_4/meaure_4: text
ingredient_n/meaure_n: text

3. Analysis

3.1. Areas, Ingredients and Categories

The meal database has a collection of meals that come from different areas/countries. The meals are split into different categories.

As an initial step, and to get a feel of the diversity of the collection, we count the number of meals per area/country.

```
# analysis pipeline 1: counting the number of meals per area

# first we group by area and count the number of meals/area
group = {"$group": {"_id": "$area", "total_number_of_meals": {"$sum":1}}}

# then we sort by number of meals
sort = {"$sort": {"total_number_of_meals":-1}}

# and project the required fields
project = {"$project": {"_id":0, "area":"$_id", "total_number_of_meals":1}}


cursor = db.meals.aggregate([group, sort, project])
df1 = pd.DataFrame(cursor)
df1.head(20)
```

	total_number_of_meals	area
0	56	British
1	32	American
2	28	French
3	18	Italian
4	13	Canadian
5	12	Chinese
6	11	Indian
7	8	Croatian
8	8	Greek
9	8	Jamaican
10	8	Polish
11	8	Malaysian
12	8	Irish
13	8	Japanese
14	8	Tunisian
15	8	Egyptian
16	8	Portuguese
17	7	Moroccan
18	5	Mexican
19	4	Dutch

Results show that the majority of the meals come from Britain, followed by the USA and then France. We also see that there is a wide range regarding the number of meals per area, this should be taken into consideration in further analysis.

Now let's see how many categories there are and the number of meals per category.

```python
# analysis pipeline 2

# first we group by catgory and count the number of meals/category
group = {"$group": {"_id": "$category", "number_of_meals": {"$sum":1}}}

# we sort by number of meals
sort = {"$sort": {"number_of_meals":-1}}

# and project the required fields
project = {"$project": {"_id":0, "category":"$_id", "number_of_meals":1}}


cursor = db.meals.aggregate([group, sort, project])
df2 = pd.DataFrame(cursor)
df2
```

	number_of_meals	category
0	64	Dessert
1	42	Beef
2	35	Chicken
3	32	Vegetarian
4	27	Seafood
5	18	Pork
6	16	Side
7	14	Lamb
8	11	Miscellaneous
9	8	Pasta
10	7	Breakfast
11	4	Starter
12	3	Vegan
13	1	Goat

The majority of meals are desserts! There is also a miscellaneous category.

Now we dig deeper. We want to find out which area/cuisine uses the most ingredients!

To make the comparison as fair as possible, we only include the following meal categories: Beef, Chicken, Seafood, Pork, Lamb and Pasta. So we are focusing on main dishes.

The goal of this analysis is to find out the average number of ingredients per meal per area.

```
# analysis pipeline 3: average number of ingredients per cuisine

# first unwind the ingredients array
unwind = {"$unwind": "$ingredients"}

# now match the chosen categories
match = {"$match": {"category": {"$in": ["Beef", "Chicken", "Seafood", 'Pork', 'Lamb', "Pasta", "Vegetarian"]}}}

# we group by area and put unique ingredients/area into a set 'ingredient' and count them
group = {"$group": {"_id": "$area",
                    "ingredient":{"$addToSet":"$ingredients"},
                    "number_of_unique_ingredients":{"$sum":1}}}

# we sort by number of unique ingredients
sort = {"$sort": {"number_of_unique_ingredients":-1}}

# and project the required output
project = {"$project": {"_id":0, "area":"$_id",
                        "set_of_unique_ingredients": "$ingredient",
                        "number_of_unique_ingredients":1}}


cursor = db.meals.aggregate([unwind, match, group, sort, project])
df3 = pd.DataFrame(cursor)
df3.head(5)
```

	number_of_unique_ingredients	area	set_of_unique_ingredients
0	262	British	[Red Onions, Egg White, Cinnamon, Honey, Lemon, Chopped Tomatoes, mushrooms, Beef Fillet, Brown Lentils, Squash, Mustard, White Wine, Vegetable Oil, eggs, Butter, Bacon, Smoked Paprika, Water, Baking Powder, Nutmeg, Green Beans, Haddock, Cocoa, O...
1	181	French	[Chicken Legs, Butter, Beef Stock, Harissa Spice, Breadcrumbs, Bouquet Garni, Plain Flour, Chicken Stock, Challots, Potatoes, Chives, Brandy, Bread, Chicken Thighs, Cannellini Beans, Tomatoes, Dry White Wine, Creme Fraiche, Cream Cheese, Thyme, T...
2	173	American	[garlic, Sugar, Apple Cider Vinegar, Pine nuts, Soy Sauce, Ground Beef, Sesame Seed Burger Buns, Barbeque Sauce, Broccoli, Minced Beef, Chicken, onion salt, oregano, Black pepper, Chicken Breast, Brown Sugar, Cayenne pepper, Whole Milk, Milk, Kos...
3	171	Chinese	[Hotsauce, Garlic Powder, Coriander, Ginger Cordial, Tomato Puree, Egg White, Vinegar, Chilli, Oil, Sunflower Oil, Chicken Stock, Onion Salt, Peanuts, Chilli Powder, Fermented Black Beans, Sichuan pepper, Plain Flour, Sugar, Garlic Clove, Corn Fl...
4	158	Italian	[Spinach, Nutmeg, Chicken Stock Cube, olive oil, Sugar, Corn Flour, Clotted Cream, Fettuccine, Red Wine Vinegar, mozzarella balls, onions, Water, Spaghetti, Lime, Bacon, onion, Mascarpone, Mozzarella, Canned tomatoes, Lasagne Sheets, pepper, From...

```
# quick data validation: checking that the set of ingredients returned is indeed unique with no duplicates

sample = df3.loc[0].set_of_unique_ingredients
sample.sort()
sample[:30]
```

```
['Apple Cider Vinegar',
 'Avocado',
 'Bacon',
 'Baking Powder',
 'Bay Leaf',
 'Bay Leaves',
 'Beef',
 'Beef Fillet',
 'Beef Stock',
 'Black Olives',
 'Breadcrumbs',
 'Broccoli',
 'Brown Lentils',
 'Butter',
 'Carrots',
 'Cashews',
 'Celery',
 'Challots',
 'Cherry Tomatoes',
 'Chestnuts',
 'Chicken',
 'Chicken Breast',
 'Chicken Stock',
 'Chicken Stock Cube',
 'Chili Powder',
 'Chopped Tomatoes',
 'Cinnamon',
 'Cocoa',
 'Cold Water',
 'Coriander']
```

Indeed no duplicates. There are, however, similar ingredients that are written differently such as chicken stock and chicken stock cube. We assume, for ease of analysis, that the percentage of such occurences is not high and that they occur in each area, meaning they cancel each other out.

Now we perform the same query as in pipeline 1, to get the total number of meals per area, but only for the categories we used in pipeline 3, ie: Beef, Chicken, Seafood, Pork, Lamb and Pasta

```
# analysis pipeline 4
# the same as pipeline 1, but we restrict the meal categories to those used in pipeline 3


# match the chosen categories
match = {"$match": {"category": {"$in": ["Beef", "Chicken", "Seafood", 'Pork', 'Lamb', "Pasta", "Vegetarian"]}}}

# group by area and count the number of meals/area
group = {"$group": {"_id": "$area", "number_of_meals": {"$sum":1}}}

# then we sort by number of meals
sort = {"$sort": {"number_of_meals":-1}}

# and project the required fields
project = {"$project": {"_id":0, "area":"$_id", "number_of_meals":1}}


cursor = db.meals.aggregate([match, group, sort, project])
df4 = pd.DataFrame(cursor)
df4.head(10)
```

	number_of_meals	area
0	22	British
1	14	American
2	14	Italian
3	13	French
4	12	Chinese
5	10	Indian
6	7	Portuguese
7	7	Jamaican
8	7	Greek
9	7	Moroccan

Now, we merge the reults of pipelines 3 and 4 as follows:

```
# merging pipelines 3 and 4
merged_pipelines = df4.merge(df3, on="area",how="inner")


# reordering columns as desired
ingredients_per_area = merged_pipelines[["area", "number_of_meals", "number_of_unique_ingredients"]]
ingredients_per_area.head(10)
```

	area	number_of_meals	number_of_unique_ingredients
0	British	22	262
1	Italian	14	158
2	American	14	173
3	French	13	181
4	Chinese	12	171
5	Indian	10	133
6	Irish	7	75
7	Egyptian	7	69
8	Japanese	7	68
9	Portuguese	7	85

To account for the discrepency in the number of meals, we calculate an average number of ingredients/meal. Areas with less than 5 meals are excluded.

```
# filtering the data frame by removing countries that have less than 5 meals
ingredients_per_area_filtered = ingredients_per_area[ingredients_per_area["number_of_meals"] >= 5]


# adding a new calculated column representing the average of ingredients/meal
ingredients_per_area_filtered["avg_number_of_ingredients_per_meal"] =\
ingredients_per_area_filtered["number_of_unique_ingredients"]/ingredients_per_area_filtered["number_of_meals"]


# showing the average number of ingredients in descending order
avg_num_of_ingredients = ingredients_per_area_filtered[["area", "avg_number_of_ingredients_per_meal"]]

avg_num_of_ingredients.sort_values(by=["avg_number_of_ingredients_per_meal"], ascending = False)
```

	area	avg_number_of_ingredients_per_meal
11	Jamaican	16.285714
4	Chinese	14.250000
3	French	13.923077
5	Indian	13.300000
12	Moroccan	12.857143
13	Malaysian	12.666667
2	American	12.357143
14	Polish	12.200000
9	Portuguese	12.142857
15	Mexican	12.000000
0	British	11.909091
1	Italian	11.285714
6	Irish	10.714286
16	Tunisian	10.200000
7	Egyptian	9.857143
10	Greek	9.714286
8	Japanese	9.714286

It was expected to see Indian, Chinese and Mexican cuisines on the top of this list, however, it seems that Jamaican recipes have the highest number of ingredients per meal!

3.2. Where are the vegetarians?

Which cuisine has the biggest proporion of vegetarian recipes? Let's find out.

```
# analysis pipeline 5

# match only vegetarian recipes
match = {"$match": {"category": "Vegetarian"}}

# group by area, and count meals
group = {"$group": {"_id": "$area", "count": {"$sum":1}}}

# sort by descending order of number of vegetarian meals
sort = {"$sort": {"count": -1}}

# project the requires fields
project = {"$project": {"_id":0, "area": "$_id", "number_of_vegetarian_meals":"$count"}}

cursor = db.meals.aggregate([match, group, sort, project])

df5 = pd.DataFrame(cursor)
df5
```

	area	number_of_vegetarian_meals
0	Egyptian	4
1	French	4
2	Indian	4
3	Italian	4
4	British	4
5	Moroccan	3
6	Tunisian	2
7	American	2
8	Greek	1
9	Mexican	1
10	Japanese	1
11	Chinese	1
12	Spanish	1

As an Egyptian myself, I was not surprised to see the Egyptian cuisine at the top of the list. But to truly evaluate how abundant vegetarian meals are in a certain cuisine, we have to view these numbers as a proportion of the total number of meals.To do that, we merge the total number of meals from pipeline 1 with the results we obtained.

```
# merging pipelines 1 and 5
merged_pipelines = df1.merge(df5, on="area",how="inner")


# reordering columns as desired
veg_meals = merged_pipelines[["area", "total_number_of_meals", "number_of_vegetarian_meals"]]


# adding a new calculated column representing the percentage of vegetarian meals per cuisine
veg_meals["percentage_of_veg_meals"] = round((veg_meals["number_of_vegetarian_meals"]/veg_meals["total_number_of_meals"])*1
veg_meals.sort_values("percentage_of_veg_meals", ascending = False)
```

	area	total_number_of_meals	number_of_vegetarian_meals	percentage_of_veg_meals
1	Egyptian	8	4	50.00
3	Moroccan	7	3	42.86
12	Indian	11	4	36.36
4	Spanish	3	1	33.33
0	Tunisian	8	2	25.00
11	Italian	18	4	22.22
5	Mexican	5	1	20.00
6	French	28	4	14.29
7	Greek	8	1	12.50
10	Japanese	8	1	12.50
9	Chinese	12	1	8.33
2	British	56	4	7.14
8	American	32	2	6.25

Again, we see the Egyptian cuisine at the top of the list, with half of its recipes being Vegetarian. The Americans seem to really like their protein!

3.3. The most used ingredient

What if we want to find out the most used ingredient for a certain cuisine?

Here, we focus on the Chinese, Italian, American, British, Indian and French cuisines.

```
# analysis pipeline 6

# unwinding the ingredients array
unwind = {"$unwind": "$ingredients"}

# Excluding desserts from categories, water from ingredients and matching the cuisines we are interested in
match = {"$match": {"category": {"$nin": ["Dessert"]},
                    "area": {"$in": ["Chinese", "Italian", "American", "British", "Indian", "French"]},
                    "ingredients": {"$nin": ["water", "Water"]}}
        }

# grouping by area and ingredient
group1 = {"$group": {"_id": {"area":"$area", "ingredients":"$ingredients"},
                     "count": {"$sum": 1}
                     }
         }

# sorting by descending order
sort = {"$sort": {"count": -1}}

# grouping again by area and singling out the ingredient with the highest count
group2 = {"$group": {
    "_id": "$_id.area",
    "most_used_ingredient_name":  {"$first": "$_id.ingredients"},
    "most_used_ingredient_count":   {"$first": "$count"}
                    }
         }

# projecting required fields
project = {"$project": {"_id":1,
                        "most_used_ingredient_count":1,
                        "most_used_ingredient_name":1
                       }
          }


cursor = db.meals.aggregate([unwind, match, group1, sort, group2, project])

df = pd.DataFrame(cursor)
df
```

	_id	most_used_ingredient_name	most_used_ingredient_count
0	Chinese	Soy Sauce	10
1	American	Butter	10
2	Italian	Olive Oil	10
3	Indian	Garlic	7
4	French	Olive Oil	12
5	British	Plain Flour	12

The most commonly used ingredient in the Chinese cuisine is Soya Sauce. Who's schocked? I'm not. French and Italian cuisines share Olive oil as the most used ingredient, and for the American Cuisine, it seems to be butter.

Which other cuisines rely on butter heavily?

3.4. Everything is better with butter!

In addition to finding out which cuisines use butter the most, we want to *how* much butter is used. This is a bit challenging given the heterogeneity by which measures are documented.

```python
# analysis pipeline 7

# matching recipes that are not dessert
match1 = {"$match": {"category": {"$nin": ["Dessert"]}}}

# first projection, we convert the recipes field into an array
project1 = {"$project": {"area":1, "recipes": {"$objectToArray": "$recipe"}, "category":1}}

# then we unwind
unwind = {"$unwind": "$recipes"}

# group by area and key values (ingredients) of the recipe field and push the amounts into an array
group = {"$group": {"_id": {"area": "$area", "key":"$recipes.k"},
                    "amounts":{"$push": "$recipes.v"}, "count": {"$sum":1}}}

# sorting by count of area/ingredient combination
sort = {"$sort": {"count":-1}}

# matching ingredients that are butter
match2 = {"$match": {"_id.key": {"$regex": "utter"}}}

# final projection of required fields
project2 = {"$project": {"_id":1, "count":1, "amounts":1}}


cursor = db.meals.aggregate([match1, project1, unwind, group, sort, match2, project2])

df7 = pd.DataFrame(cursor)
df7.head(5)
```

	_id	amounts	count
0	{'area': 'French', 'key': 'Butter'}	[180g, 30g, 25g, 50g, 60g, For Greasing, 2 knobs, 2 tbs, 50g]	9
1	{'area': 'British', 'key': 'Butter'}	[1 knob, 25g, 25g, 250g, 1 knob, 200g, 100g , 75g]	8
2	{'area': 'American', 'key': 'Butter'}	[50g, 4 tablespoons (55 grams), 2 1/2 Tbs, To serve, 2 tbsp, 2 tbsp]	6
3	{'area': 'Irish', 'key': 'Butter'}	[3 tbs, Knob, 50g, 50g]	4
4	{'area': 'Polish', 'key': 'Butter'}	[2 tbs, 3 tbs, 6 tblsp]	3

The British and French cuisines seem to use butter often as well. We try to experiment with the amounts in the British cuisine, and get an estimate of the average amount of butter used per meal.

```python
# singling out the British cuisine
butter_amounts_british = df7.loc[1].amounts

# extracting exact amounts in grams
amounts_in_grams = []

for i in butter_amounts_british:
    if 'g' in list(i):
        amounts_in_grams.append(i)

print(*amounts_in_grams)
```

```
25g 25g 250g 200g 100g  75g
```

Now we need these values as actual integers

```python
int_amounts_in_grams = []

for i in amounts_in_grams:
    list_amount =[int(x) for x in list(i) if x.isdigit()]
    int_amount = int("".join(map(str,list_amount)))
    int_amounts_in_grams.append(int_amount)

print(*int_amounts_in_grams)
```

```
25 25 250 200 100 75
```

```python
sum(int_amounts_in_grams)/len(int_amounts_in_grams)
```

```
112.5
```

112 grams on average in non-dessert meals, hmmm!

I am curious to know which recipe uses 250 grams of butter. Let's search for it.

```
# analysis pipeline 8

project1 = {"$project": {"recipes": {"$objectToArray": "$recipe"}, "recipe":1, "area":1, "category":1, "meal_name":1}}

unwind = {"$unwind": "$recipes"}

match = {"$match": {"area": "British",
         "category": {"$nin": ["Dessert"]},
         "recipes.k": {"$regex": "utter"},
         "recipes.v": {"$regex": "g"}}}

project2 = {"$project": {"_id":0, "meal_name":1, "recipe":1}}

cursor = db.meals.aggregate([project1, unwind, match, project2])

df8 = pd.DataFrame(cursor)
for i in range(1, len(df8)):
    print(df8.loc[i].meal_name)
    print(df8.loc[i].recipe)
    print(50*("*"))
```

```
Beef and Mustard Pie
{'Beef': '1kg', 'Plain Flour': '2 tbs', 'Rapeseed Oil': '2 tbs', 'Red Wine': '200ml', 'Beef Stock': '400ml', 'On
ion': '1 finely sliced', 'Carrots': '2 chopped', 'Thyme': '3 sprigs', 'Mustard': '2 tbs', 'Egg Yolks': '2 free-r
ange', 'Puff Pastry': '400g', 'Green Beans': '300g', 'Butter': '25g', 'Salt': 'pinch', 'Pepper': 'pinch'}
**************************************************
Beef Dumpling Stew
{'Olive Oil': '2 tbs', 'Butter': '25g', 'Beef': '750g', 'Plain Flour': '125g', 'Garlic': '2 cloves minced', 'Oni
ons': '175g', 'Celery': '150g', 'Carrots': '150g', 'Leek': '2 chopped', 'Swede': '200g', 'Red Wine': '150ml', 'B
eef Stock': '500g', 'Bay Leaf': '2', 'Thyme': '3 tbs', 'Parsley': '3 tblsp chopped', 'Baking Powder': '1 tsp ',
'Suet': '60g', 'Water': 'Splash'}
**************************************************
Beef and Oyster pie
{'Beef': '900g', 'Olive Oil': '3 tbs', 'Shallots': '3', 'Garlic': '2 cloves minced', 'Bacon': '125g', 'Thyme':
'1 tbs chopped', 'Bay Leaf': '2', 'Stout': '330ml', 'Beef Stock': '400ml', 'Corn Flour': '2 tbs', 'Oysters':
'8', 'Plain Flour': '400g', 'Salt': 'pinch', 'Butter': '250g', 'Eggs': 'To Glaze'}
**************************************************
Chicken Ham and Leek Pie
{'Chicken Stock': '450ml', 'Chicken Breast': '3', 'Butter': '200g', 'Leek': '2 sliced', 'Garlic': '2 cloves minc
ed', 'Plain Flour': '350g', 'Milk': '200ml', 'White Wine': '3 tbs', 'Double Cream': '150ml', 'Ham': '150g', 'Sea
Salt': 'spinkling', 'Pepper': 'pinch', 'Free-range Egg, Beaten': '1', 'Cold Water': '1 tbls'}
**************************************************
Lancashire hotpot
{'Butter': '100g ', 'Lamb': '900g', 'Lamb Kidney': '3', 'Onions': '2 medium', 'Carrots': '4 sliced', 'Plain Flou
r': '25g', 'Worcestershire Sauce': '2 tsp', 'Chicken Stock': '500ml', 'Bay Leaves': '2', 'Potatoes': '900g'}
**************************************************
Three Fish Pie
{'Potatoes': '1kg', 'Butter': '75g', 'Milk': '568ml', 'Gruyère': '50g', 'Leek': '2 sliced', 'Plain Flour': '75
g', 'White Wine': '150ml', 'Parsley': '2 tbs chopped', 'Salmon': '250g', 'Haddock': '250g', 'Smoked Haddock': '2
50g', 'Eggs': '6'}
**************************************************
```

Found it! The Beef and Oyster Pie. Well, at least it's for 900 grams of Beef!

There's also another pie that uses 200 grams of butter, The Chicken and Ham Pie! And a third pie that has only 75 grams of butter, but makes up for it with 50 grams of Gruyère! The British cuisine is full of surprises (and pies).

4. Conclusions

NoSQL databases such as MongoDB offer incredible flexibility in storing, transforming and querying data. However, the data structure plays a role as well, in the feasibility of querying and reaching the required information.

This is experienced first-hand in this project, as the original data structure was far from ideal. However, MongoDB's transformation pipelines offer a variety of handy tools that transformed the data completely in a few lines of code. Visualizing the data and monitoring data transformation using MongoDB compass, made it an even friendlier experience. The user interface of MongoDB compass is simple, efficient and incredibly user-friendly.

Personally, I both learned a lot and had fun experimenting with this API's data. As an aspiring data scientist, I find it very important to learn how to deal with different data structures, and contrary to my initial belief, dealing with document-based data is intuitive and at times simpler than querying relational databases, especially those that have complex relationships.