The distinct() method

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

datacamp

An exceptional laureate

```
db.laureates.find_one({"prizes.2": {"$exists": True}})
```

```
{'_id': ObjectId('5baacf97f35b632bbe12c1ad'),
 'born': '0000-00-00',
 'died': '0000-00-00',
 'firstname': ('Comité international de la Croix Rouge'
               '(International Committee of the Red Cross)'),
 'gender': 'org',
 'id': '482',
 'prizes': [{'affiliations': [[]],
   'category': 'peace',
   'share': '1',
   'year': '1917'},
  {'affiliations': [[]], 'category': 'peace',
   'share': '1', 'year': '1944'},
  {'affiliations': [[]], 'category': 'peace',
   'share': '2', 'year': '1963'}]}
```

Using .distinct()

```
db.laureates.distinct("gender")
```

```
['male', 'female', 'org']
```

- A convenience method for a common aggregation (like `count_documents`)

- We will learn how to create custom aggregations in Chapter 4

- `distinct` aggregation is efficient if there is a collection *index* on the field

- We will learn how to create an index in Chapter 3

- No index needed here: collection fits in memory, has ?1,000 documents

.distinct() with dot notation

```
db.laureates.find_one({"prizes.2": {"$exists": True}})
```

```
db.laureates.distinct("prizes.category")
```

```
{'_id': ObjectId('5baacf97f35b632bbe12c1ad'),
 'born': '0000-00-00',
 'died': '0000-00-00',
 'firstname': ('Comité international de la Croix Rouge'
                '(International Committee of the Red Cross)'),
 'gender': 'org',
 'id': '482',
 'prizes': [{'affiliations': [[]],
   'category': 'peace',
   'share': '1',
   'year': '1917'},
  {'affiliations': [[]], 'category': 'peace',
   'share': '1', 'year': '1944'},
  {'affiliations': [[]], 'category': 'peace',
   'share': '2', 'year': '1963'}]}
```

```
['physics', 'chemistry', 'peace',
  'medicine', 'literature', 'economics']
```

```
['physics', 'chemistry', 'peace', 'medicine', 'literature', 'economi
```

Let's practice!

INTRODUCTION TO MONGODB IN PYTHON

Pre-filtering distinct values

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

Awards into prize shares

```python
db.laureates.find_one({"prizes.share": "4"})
```

```python
{'bornCountry': 'France',
 'died': '1906-04-19',
 'diedCountry': 'France',,
 'firstname': 'Pierre',
 'prizes': [{'affiliations': [{'city': 'Paris',
   'country': 'France',
   'name': ('École municipale de physique et de chimie'
            'industrielles (Municipal School of Industrial'
            'Physics and Chemistry)')}],
  'motivation': ('"in recognition of the extraordinary'
                 'services they have rendered by their'
                 'joint researches on the radiation'
                 'phenomena discovered by Professor'
                 'Henri Becquerel"'),
  'category': 'physics', 'share': '4', 'year': '1903'}],
 'surname': 'Curie',
 ...
}
```

High-share prize categories

```
db.laureates.distinct("prizes.category")
```

```
['physics', 'chemistry', 'peace',
 'medicine', 'literature', 'economics']
```

```
list(db.laureates.find({"prizes.share": "4"}))
```

```
[...]
```

```
db.laureates.distinct(
    "prizes.category", {"prizes.share": '4'})
```

```
['physics', 'chemistry', 'medicine']
```

```
db.prizes.distinct("category", {"laureates.share": "4"})
```

```
['physics', 'medicine', 'chemistry']
```

Prize categories with multi-winners

```
db.laureates.count_documents({"prizes.1": {"$exists": True}})
```

```
6
```

```
db.laureates.distinct(
    "prizes.category", {"prizes.1": {"$exists": True}})
```

```
['chemistry', 'physics', 'peace']
```

```
# We'll learn how to do this in the next chapter:
[[{'category': 'physics'}, {'category': 'chemistry'}],
 [{'category': 'physics'}, {'category': 'physics'}],
 [{'category': 'chemistry'}, {'category': 'peace'}],
 [{'category': 'chemistry'}, {'category': 'chemistry'}],
 [{'category': 'peace'}, {'category': 'peace'},
  {'category': 'peace'}],
 [{'category': 'peace'}, {'category': 'peace'}]]
```

Practice time!

INTRODUCTION TO MONGODB IN PYTHON

Array fields and equality

```
{'firstname': 'John',
 'surname': 'Bardeen',
 'prizes': [{
    'category': 'physics', 'year': '1956', 'share': '3',
    'motivation': ('"for their researches on semiconductors and their'
                   'discovery of the transistor effect"'),
    ...
  }, {
    'category': 'physics', 'year': '1972', 'share': '3',
    'motivation': ('"for their jointly developed theory of'
                   'superconductivity, usually called the BCS-theory"'),
    ...
  }],
 ...
}
```

```
db.laureates.count_documents({"prizes.category": "physics"})
```

```
206
```

Array fields and equality, simplified

```
# Imaginary extra field in John Bardeen's document:
{"nicknames": ["Johnny", "JSwitch", "JB". "Tc Johnny", "Bardy"]}
```

```
db.laureates.find({"nicknames": "JB"})
# different than {"nicknames": ["JB"]}
```

Array fields and operators

```
db.laureates.count_documents(
    {"prizes.category": "physics"})
```

```
206
```

```
db.laureates.count_documents(
    {"prizes.category": {"$ne": "physics"}})
```

```
716
```

```
db.laureates.count_documents({
        "prizes.category": {
            "$in": ["physics", "chemistry", "medicine"]}})
```

```
596
```

```
db.laureates.count_documents({
        "prizes.category": {
            "$nin": ["physics", "chemistry", "medicine"]}})
```

```
326
```

```
326
```

Enter $elemMatch

```
db.laureates.count_documents({
    "prizes": {
      "category": "physics", "share": "1"}})
```

```
0
```

```
db.laureates.count_documents({
    "prizes.category": "physics", "prizes.share": "1"})
```

```
48
```

```
db.laureates.count_documents({
    "prizes": {"$elemMatch":
                {"category": "physics", "share": "1"}}})
```

```
47
```

```
db.laureates.count_documents({
    "prizes": {"$elemMatch": {
        "category": "physics",
        "share": "1",
        "year": {"$lt": "1945"},}}})
```
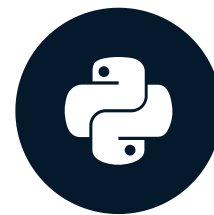
```
29
```

```
29
```

Onward and array-ward!

INTRODUCTION TO MONGODB IN PYTHON

Distinct As You Like It: Filtering with Regular Expressions

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

Finding a substring with $regex

```python
db.laureates.find_one({"firstname": "Marie"})
```

```python
{'born': '1867-11-07',
 'bornCity': 'Warsaw',
 'bornCountry': 'Russian Empire (now Poland)',
 'firstname': 'Marie',
 'surname': 'Curie, née Sklodowska',
 ...}
```

```python
db.laureates.distinct("bornCountry",
                      {"bornCountry": {"$regex": "Poland"}})
```

```python
['Russian Empire (now Poland)',
 'Prussia (now Poland)',
 'Germany (now Poland)',
 'Austria-Hungary (now Poland)',
 'German-occupied Poland (now Poland)',
 'Poland',
 'Poland (now Ukraine)',
 'Poland (now Lithuania)',
 'Poland (now Belarus)',
 'Free City of Danzig (now Poland)']
```

Flag options for regular expressions

```python
case_sensitive = db.laureates.distinct(
    "bornCountry",
    {"bornCountry": {"$regex": "Poland"}})
```

```python
case_insensitive = db.laureates.distinct(
    "bornCountry",
    {"bornCountry": {"$regex": "poland", "$options": "i"}})

assert set(case_sensitive) == set(case_insensitive)
from bson.regex import Regex

db.laureates.distinct("bornCountry",
                      {"bornCountry": Regex("poland", "i")})
```

```
['Russian Empire (now Poland)', ...]
```

```python
import re


db.laureates.distinct("bornCountry",
                      {"bornCountry": re.compile("poland", re.I)})
```

```
['Russian Empire (now Poland)', ...]
```

Beginning and ending (and escaping)

```
from bson.regex import Regex

db.laureates.distinct("bornCountry",
                      {"bornCountry": Regex("^Poland")})
```

```
['Poland',
 'Poland (now Ukraine)',
 'Poland (now Lithuania)',
 'Poland (now Belarus)']
```

```
db.laureates.distinct(
    "bornCountry",
    {"bornCountry": Regex("^Poland \(now")})
```

```
['Poland (now Ukraine)',
 'Poland (now Lithuania)',
 'Poland (now Belarus)']
```

```
db.laureates.distinct(
    "bornCountry",
    {"bornCountry": Regex("now Poland\)$")})
```

```
['Russian Empire (now Poland)',
 'Prussia (now Poland)',
 'Germany (now Poland)',
 'Austria-Hungary (now Poland)',
 'German-occupied Poland (now Poland)',
 'Free City of Danzig (now Poland)']
```

Let's practice!

datacamp