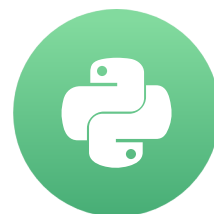# Intro to Aggregation: From Query Components to Aggregation Stages

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston

Instructor

Queries have implicit stages

```python
cursor = db.laureates.find(
    filter={"bornCountry": "USA"},
    projection={"prizes.year": 1},
    limit=3
)
for doc in cursor:
    print(doc["prizes"])
```

```python
cursor = db.laureates.aggregate([
    {"$match": {"bornCountry": "USA"}},
    {"$project": {"prizes.year": 1}},
    {"$limit": 3}
])
for doc in cursor:
    print(doc["prizes"])
```

```
[{'year': '1923'}]
[{'year': '1927'}]
[{'year': '1936'}]
```

```
[{'year': '1923'}]
[{'year': '1927'}]
[{'year': '1936'}]
```

```python
cursor = db.laureates.aggregate([
    stage_1,
    stage_2,
    ...
])
```

Adding sort and skip stages

```python
from collections import OrderedDict

list(db.laureates.aggregate([
    {"$match": {"bornCountry": "USA"}},
    {"$project": {"prizes.year": 1, "_id": 0}},
    {"$sort": OrderedDict([("prizes.year", 1)])},
    {"$skip": 1},
    {"$limit": 3}
]))
```

```
[{'prizes': [{'year': '1912'}]},
 {'prizes': [{'year': '1914'}]},
 {'prizes': [{'year': '1919'}]}]
```

But can I count?

```
list(db.laureates.aggregate([
    {"$match": {"bornCountry": "USA"}},
    {"$count": "n_USA-born-laureates"}
]))
```

```
[{'n_USA-born-laureates': 269}]
```

```
db.laureates.count_documents({"bornCountry": "USA"})
```

```
269
```

What about `db.laureates.distinct("bornCountry")` ?

Let's practice!

DataCamp

Back to Counting:

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston

Instructor

Field paths

- *expression object* ?

```
{field1: <expression1>, ...}
```

```
db.laureates.aggregate([
    {"$project": {"prizes.share": 1}}
]).next()
```

```
{'_id': ObjectId('5bd3a610053b1704219e19d4'),
                'prizes': [{'share': '1'}]}
```

- expression: `1`

```
db.laureates.aggregate([
    {"$project": {"n_prizes": {"$size": "$prizes"}}}
]).next()
```

```
{'_id': ObjectId('5bd3a610053b1704219e19d4'),
                'n_prizes': 1}
```

- expression: `{"$size": "$prizes"}`

- field path: `$prizes`

Operator expressions

```
db.laureates.aggregate([
    {"$project": {"n_prizes": {"$size": "$prizes"}}}
]).next()
```

```
{'_id': ObjectId('5bd3a610053b1704219e19d4'), 'n_prizes': 1}
```

- *operator* expression: `{"$size": "$prizes"}`

- field path: `$prizes`

```
db.laureates.aggregate([
    {"$project": {"n_prizes": {"$size": ["$prizes"]}}}
]).next()
```

```
{'_id': ObjectId('5bd3a610053b1704219e19d4'), 'n_prizes': 1}
```

One more example: a multi-parameter operator

```
db.laureates.aggregate([
    {"$project": {"solo_winner": {"$in": ["1", "$prizes.share"]}}}
]).next()
```

```
{'_id': ObjectId('5bd3a610053b1704219e19d4'), 'solo_winner': True}
```

Implementing .distinct()

```python
list_1 = db.laureates.distinct("bornCountry")
```

```python
list_2 = [doc["_id"] for doc in db.laureates.aggregate([
    {"$group": {"_id": "$bornCountry"}}
])]
set(list_2) - {None} == set(list_1)
```

```
True
```

- `$group` must map `_id`, which must be unique (like any Mongo document)
- No `$match` before `$group`
  - All distinct "bornCountry" values captured
  - including "no value" (`None`)

How many prizes have been awarded in total?

```
list(db.laureates.aggregate([
    {"$project": {"n_prizes": {"$size": "$prizes"}}},
    {"$group": {"_id": None, "n_prizes_total": {"$sum": "$n_prizes"}}}
]))
```

```
[{'_id': None, 'n_prizes_total': 941}]
```

- `{"_id": None}` ? one document out.

- `$sum` operator acts as *accumulator* in `$group` stage

Let's practice!

INTRODUCTION TO MONGODB IN PYTHON

DataCamp

Zoom into Array Fields with $unwind

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

Sizing and summing

```python
list(db.prizes.aggregate([
    {"$project": {"n_laureates": {"$size": "$laureates"},
                  "year": 1, "category": 1, "_id": 0}}
]))
```

```python
[{'year': '2018', 'category': 'physics', 'n_laureates': 3},
 {'year': '2018', 'category': 'chemistry', 'n_laureates': 3},
 {'year': '2018', 'category': 'medicine', 'n_laureates': 2},
 ...]
```

```python
list(db.prizes.aggregate([
    {"$project": {"n_laureates": {"$size": "$laureates"},
                  "category": 1}},
    {"$group": {"_id": "$category", "n_laureates":
                {"$sum": "$n_laureates"}}},
    {"$sort": {"n_laureates": -1}},
]))
```

```python
[{'_id': 'medicine', 'n_laureates': 216},
 {'_id': 'physics', 'n_laureates': 210},
 {'_id': 'chemistry', 'n_laureates': 181},
 {'_id': 'peace', 'n_laureates': 133},
 {'_id': 'literature', 'n_laureates': 114},
 {'_id': 'economics', 'n_laureates': 81}]
```

How to $unwind

```
list(db.prizes.aggregate([
    {"$unwind": "$laureates"},
    {"$project": {
        "_id": 0, "year": 1, "category": 1,
        "laureates.surname": 1, "laureates.share": 1}},
    {"$limit": 3}
]))
```

```
[{'year': '2018',
  'category': 'physics',
  'laureates': {'surname': 'Ashkin', 'share': '2'}},
 {'year': '2018',
  'category': 'physics',
  'laureates': {'surname': 'Mourou', 'share': '4'}},
 {'year': '2018',
  'category': 'physics',
  'laureates': {'surname': 'Strickland', 'share': '4'}}]
```

Renormalization, anyone?

```python
list(db.prizes.aggregate([
    {"$unwind": "$laureates"},
    {"$project": {"year": 1, "category": 1, "laureates.id": 1}},
    {"$group": {"_id": {"$concat": ["$category", ":", "$year"]},
                "laureate_ids": {"$addToSet": "$laureates.id"}}},
    {"$limit": 5}
]))
```

```
[{'_id': 'medicine:1901', 'laureate_ids': ['293']},
 {'_id': 'peace:1902', 'laureate_ids': ['465', '464']},
 {'_id': 'physics:1902', 'laureate_ids': ['3', '2']},
 {'_id': 'peace:1903', 'laureate_ids': ['466']},
 {'_id': 'medicine:1903', 'laureate_ids': ['295']}]
```

$unwind and count 'em, one by one

```python
list(db.prizes.aggregate([
    {"$project": {"n_laureates": {"$size": "$laureates"}, "category": 1}},
    {"$group": {"_id": "$category", "n_laureates": {"$sum": "$n_laureates"}}},
    {"$sort": {"n_laureates": -1}},
]))
```

```python
list(db.prizes.aggregate([
    {"$unwind": "$laureates"},
    {"$group": {"_id": "$category", "n_laureates": {"$sum": 1}}},
    {"$sort": {"n_laureates": -1}},
]))
```

```
[{'_id': 'medicine', 'n_laureates': 216},
 {'_id': 'physics', 'n_laureates': 210},
 {'_id': 'chemistry', 'n_laureates': 181},
 {'_id': 'peace', 'n_laureates': 133},
 {'_id': 'literature', 'n_laureates': 114},
 {'_id': 'economics', 'n_laureates': 81}]
```

$lookup

```python
list(db.prizes.aggregate([
    {"$match": {"category": "economics"}},
    {"$unwind": "$laureates"},
    {"$lookup": {"from": "laureates", "foreignField": "id",
                 "localField": "laureates.id", "as": "laureate_bios"
```

```python
    {"$unwind": "$laureate_bios"},
    {"$group": {"_id": None,
                "bornCountries":
                {"$addToSet": "$laureate_bios.bornCountry"}
    }},
]))
```

```python
[{'_id': None,
  'bornCountries': [
    'the Netherlands', 'British West Indies (now Saint Lucia)', 'Ita
    'Germany (now Poland)', 'Hungary', 'Austria', 'India', 'USA',
    'Canada', 'British Mandate of Palestine (now Israel)', 'Norway',
    'Russian Empire (now Russia)', 'Russia', 'Finland', 'Scotland',
    'France', 'Sweden', 'Germany', 'Russian Empire (now Belarus)',
    'United Kingdom', 'Cyprus'
```
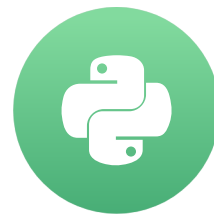
```python
bornCountries = db.laureates.distinct(
    "bornCountry", {"prizes.category": "economics"})
assert set(bornCountries) == set(agg[0]['bornCountries'])
```

Time to unwind...
with exercises!

Something Extra: $addFields to Aid Analysis

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

A somber $project

```
docs = list(db.laureates.aggregate([
    {"$project": {"died": {"$dateFromString": {"dateString": "$died"}},
                  "born": {"$dateFromString": {"dateString": "$born"}}}}
]))
```

```
OperationFailure: Error parsing date string '0000-00-00';
                  11: The parsed date was invalid ''
```

```
docs = list(db.laureates.aggregate([
    {"$match": {"died": {"$gt": "1700"}, "born": {"$gt": "1700"}}},
    {"$project": {"died": {"$dateFromString": {"dateString": "$died"}},
                  "born": {"$dateFromString": {"dateString": "$born"}}}}
]))
```

```
OperationFailure: Error parsing date string '1898-00-00';
                  11: The parsed date was invalid ''
```

split and cond-itionally correct (with $concat)

```
docs = list(db.laureates.aggregate([

    {"$match": {"died": {"$gt": "1700"}, "born": {"$gt": "1700"}}},
    {"$addFields": {"bornArray": {"$split": ["$born", "-"]},
                    "diedArray": {"$split": ["$died", "-"]}}},

    {"$addFields": {"born": {"$cond": [
        {"$in": ["00", "$bornArray"]},
        {"$concat": [{"$arrayElemAt": ["$bornArray", 0]}, "-01-01"]},
        "$born"
    ]}}},

    {"$project": {"died": {"$dateFromString": {"dateString": "$died"}},
                  "born": {"$dateFromString": {"dateString": "$born"}},
                  "_id": 0}}

]))
```

A $bucket list

```python
docs = list(db.laureates.aggregate([
    ...,
    {"$project": {"died": {"$dateFromString": {"dateString": "$died"}},
                  "born": {"$dateFromString": {"dateString": "$born"}}}},


    {"$project": {"years": {"$floor": {"$divide": [
        {"$subtract": ["$died", "$born"]},
        31557600000 # 1000 * 60 * 60 * 24 * 365.25
    ]}}}},


    {"$bucket": {"groupBy": "$years",
                 "boundaries": list(range(30, 120, 10))}}
]))


for doc in docs: print(doc)
```

```
{'_id': 30, 'count': 1}
{'_id': 40, 'count': 6}
```

Practice $addFields

INTRODUCTION TO MONGODB IN PYTHON

Wrap-Up

INTRODUCTION TO MONGODB IN PYTHON

Donny Winston
Instructor

DataCamp

You know know how to...

- Create and compose query filters and use operators

- Use dot notation

- Fetch values, arrays, use regex

- Project, sort, index

- Aggregate

- **MongoDB documentation**

- **PyMongo documentation**

Thanks!

INTRODUCTION TO MONGODB IN PYTHON