MongoDB Aggregation Examples

- Ensure you have pymongo installed before running cells in this notebook

In [ ]:
```
import pymongo
from bson.json_util import dumps
import pprint

#  -->  Update the URI with your username and password <--

uri = "mongodb://kenneth:123@localhost:27017/"
client = pymongo.MongoClient(uri)

from pymongo import MongoClient
client = MongoClient("mongodb://kenneth:123@27017/mflix")
mflixdb = client['mflix']

# mflixdb = client.mflix
demodb = client.demodb
```

About Aggregates in PyMongo

- Aggregation uses *pipelines*.
- A **pipeline** is a sequence of stages through which documents proceed.
- Some of the different stages that can be used are:
  - match
  - project
  - sort
  - limit
  - unwind
  - group
  - lookup

$match

In [ ]:
```
c = mflixdb.movies.aggregate([
    {"$match": {"year": {"$lte": 1920}}},
])

print(dumps(c, indent=4))
```

match and project

In [ ]:
```
c = mflixdb.movies.aggregate([
    {"$match": {"year": {"$lte": 1920}}},
    {"$project": {"_id":0, "title": 1, "cast": 1}},
])

print(dumps(c, indent=4))
```

match project limit and sort

In [ ]:
```
c = mflixdb.movies.aggregate([
    {"$match": {"year": {"$lte": 1920}}},
    {"$sort": {"title": 1}},
    {"$limit": 5},
    {"$project": {"_id":0, "title": 1, "cast": 1}},
])

print(dumps(c, indent=4))
```

Unwind

In [ ]:
```
c = mflixdb.movies.aggregate([
    {"$match": {"year": {"$lte": 1920}}},
    {"$sort": {"imdb.rating": -1}},
    {"$limit": 5},
    {"$unwind": "$cast"},
    {"$project": {"_id":0, "title": 1, "cast": 1, "rating": "$imdb.rating"}},
])

print(dumps(c, indent=4))
```

Grouping

In [ ]:
```
# What is the average IMDB rating of all movies by year? sort the data by year.

# like groupby in sql
c = mflixdb.movies.aggregate([
    {"$group": {"_id": {"release year": "$year"}, "Avg Rating": {"$avg": "$imdb.rat
    {"$sort" : {"_id": 1}}

])
print(dumps(c, indent = 2))
```

In [ ]:
```
# What is the average IMDB rating of all movies by year? sort the data by avg ratin

c = mflixdb.movies.aggregate([
    {"$group": {"_id": {"release year": "$year"}, "Avg Rating": {"$avg": "$imdb.rat
    {"$sort" : {"Avg Rating": -1, "_id": 1}}

])
print(dumps(c, indent = 2))
```

Lookup

In [ ]:
```
data = demodb.customers.aggregate([
    {
        "$lookup": {
            "from": "orders",
            "localField": "custid",
            "foreignField": "custid",
            "as": "orders"
        }
    },
    {"$project": {"_id": 0, "address": 0}}
])
# hardcoding stages of aggregate limits flexibility
# can make a dictionary for each stage, then provide order of stages

print(dumps(data, indent = 2))
```

Reformatting Queries

In [ ]:
```
match = {"$match": {"year": {"$lte": 1920}}}
limit = {"$limit": 5}
project = {"$project": {"_id":0, "title": 1, "cast": 1, "rating": "$imdb.rating"}}

agg = mflixdb.movies.aggregate([match, limit, project])
print(dumps(agg, indent=2))
```

In [ ]:
```

```