MongoDB + PyMongo Example Queries

- Make sure your MongoDB container is running
- Make sure you have pymongo installed before running cells in this notebook. If not, use `pip install pymongo`.

In [1]:
```
import pymongo
from bson.json_util import dumps

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

uri = "mongodb://kenneth:123@localhost:27018"
client = pymongo.MongoClient(uri)
mflixdb = client.mflix
demodb = client.demodb
```

In [ ]:
```
# Setup DemoDB with 2 collections
demodb.customers.drop()
demodb.orders.drop()

customers = [
   {"custid": "C13", "name": "T. Cruise", "address": { "street": "201 Main St.", "c
   {"custid": "C25", "name": "M. Streep", "address": { "street": "690 River St.", "
   {"custid": "C31", "name": "B. Pitt", "address": { "street": "360 Mountain Ave.",
   {"custid": "C35", "name": "J. Roberts", "address": { "street": "420 Green St.",
   {"custid": "C37", "name": "T. Hanks", "address": { "street": "120 Harbor Blvd.",
   {"custid": "C41", "name": "R. Duvall", "address": { "street": "150 Market St.",
   {"custid": "C47", "name": "S. Loren", "address": { "street": "Via del Corso", "c
]

orders = [
   { "orderno": 1001, "custid": "C41", "order_date": "2017-04-29", "ship_date": "20
   { "orderno": 1002, "custid": "C13", "order_date": "2017-05-01", "ship_date": "20
   { "orderno": 1003, "custid": "C31", "order_date": "2017-06-15", "ship_date": "20
   { "orderno": 1004, "custid": "C35", "order_date": "2017-07-10", "ship_date": "20
   { "orderno": 1005, "custid": "C37", "order_date": "2017-08-30", "items": [ { "it
   { "orderno": 1006, "custid": "C41", "order_date": "2017-09-02", "ship_date": "20
   { "orderno": 1007, "custid": "C13", "order_date": "2017-09-13", "ship_date": "20
   { "orderno": 1008, "custid": "C13", "order_date": "2017-10-13", "items": [ { "it
]

demodb.customers.insert_many(customers)
demodb.orders.insert_many(orders)

numCustomers = demodb.customers.count_documents({})
numOrders = demodb.orders.count_documents({})

print(f'There are {numCustomers} customers and {numOrders} orders')
```

In [ ]:
```
# The key (_id) attribute is automatically returned unless you explicitly say to re

# SELECT name, rating FROM customers
data = demodb.customers.find({}, {"name":1, "rating":1})
print(dumps(data, indent=2))
```

In [ ]:
```
# Now without the _id field.

# SELECT name, rating FROM customers
data = demodb.customers.find({}, {"name":1, "rating":1, "_id":0})
print(dumps(data, indent=2))
```

All fields EXCEPT specific ones returned

In [ ]:
```
# For every customer, return all fields except _id and address.

data = demodb.customers.find({}, {"_id": 0, "address": 0})
print(dumps(data, indent=2))
```

Equivalent to SQL LIKE operator

In [ ]:
```
# SELECT name, rating FROM customers WHERE name LIKE 'T%'

# Regular Expression Explanation:
  # ^ - match beginning of line
  # T - match literal character T (at the beginning of the line in this case)
  # . - match any single character except newline
  # * - match zero or more occurrences of the previous character (the . in this cas

data = demodb.customers.find({"name": {"$regex": "^T.*"}}, {"_id": 0, "name": 1, "r
print(dumps(data, indent=2))
```

Sorting and limiting

In [ ]:
```
# SELECT name, rating FROM customers ORDER BY rating LIMIT 2

data = demodb.customers.find( { }, {"_id": 0, "name": 1, "rating":1} ).sort("rating
print(dumps(data, indent=2))
```

In [ ]:
```
# Same as above, but sorting in DESC order

# SELECT name, rating FROM customers ORDER BY rating DESC LIMIT 2

data = demodb.customers.find( { }, {"_id": 0, "name": 1, "rating":1} ).sort("rating
print(dumps(data, indent=2))
```

In [ ]:
```
# Providing 2 sort keys...
```

```
data = demodb.customers.find( { }, {"_id": 0, "name": 1, "rating":1} ).sort({"ratin
print(dumps(data, indent=2))
```

Your Turn with mflix DB

Question 1

In [ ]:
```
# How many Users are there in the mflix database? How many movies?
numUsers = mflixdb.users.count_documents({})
print(f'Number of users: {numUsers}')

numMovies = mflixdb.movies.count_documents({})
print(f'Number of movies: {numMovies}')
```

Question 2

In [ ]:
```
# Which movies have a rating of “TV-G”? Only return the Title and Year.
data = mflixdb.movies.find({"rated": "TV-G"}, {"_id":0, "title":1, "year":1})
print(dumps(data, indent=2))
```

Question 3

In [ ]:
```
# Which movies have a runtime of less than 20 minutes?  Only return the title and r
data = mflixdb.movies.find({"runtime": {"$lt": 20}}, {"_id":0, "title":1, "runtime"
print(dumps(data, indent=2))
```

Question 4

In [ ]:
```
# How many theaters are in MN or MA?
numtheaters = mflixdb.theaters.count_documents({"location.address.state": {"$in": [
print(f'Number of theaters: {numtheaters}')
```

Question 5

In [ ]:
```
# Give the names of all movies that have no comments yet. Make sure the names are i
data = mflixdb.movies.find({"num_mflix_comments": 0}, {"_id":0, "title":1}).sort("t

commented_movies = mflixdb.comments.distinct("movie_id")
movies_no_comments = mflix

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

Question 6

In [ ]:
```
# Return a list of movie titles and all actors from any movie with a title that con
# Sort the list by title.
data = mflixdb.movies.find({"title": {"$regex": "Four", "$options": 1}}, {"_id":0,
print(dumps(data, indent=2))
```