DS2 MONGODB ASSIGNMENT

5조

1. Grading

- Main 에서 argument를 input으로 받아 각각 함수를 실행시키기 위해 if문 조건 추가.

- grades.json file import를 위해 import os추가 후 os.system으로 mongo import.

- 실행할 때마다 새로운 Data로 reset을 위한 collection drop function추가

* 1. 학생 성적 10개 출력하는 pagination함수 구현
     1. 실행방법 및 실행 code
        + sid와 grades만을 출력하기 위해 projection 이용하여 result cursor생성
        + sid를 ascending으로 정렬하고 10개 skip하고 10개까지 출력

result = db.grades.find({},{'\_id':0, 'note':0}).sort([('sid',1)]).limit(10).skip(10)

* + 1. 사용 MongoDB함수 및 option
       - find(projection) / sort(ascending) / limit / skip
  1. 총점과 등급을 계산하는 letter함수 구현
     1. 실행방법 및 실행 code
        + 전체 문서를 문제에 주어진 수식 대로 총점과 등급을 계산하고 update\_one을 이용하여 total과 letter field생성.

db.grades.update\_one({'sid':sid\_d['sid']}, {'$set' : {'letter':let, 'total':tot}})

* + - * total을 descending으로 정렬하고 letter와 sid와 total만 보이게 하기 위해 projection이용하여 result cursor생성.

result = db.grades.find({},{'\_id':0, 'letter':1,'sid':1, 'total':1}).sort([('total', -1)])

* + 1. 사용 MongoDB함수 및 option
       - update\_one($set) / find(projection) / sort(descending)
  1. note field및 exam = 100인 학생들에게 가산 점 추가, relative collection추가 생성, 총점 및 등급 계산하는 perfect함수 구현
     1. 실행방법 및 실행 code
        + 전체 문서를 문제에 주어진 수 식대로 총점 계산하고 update\_one을 이용하여 total과 field생성.

db.grades.update\_one({'sid':sid\_d['sid']}, {'$set' : {'total':tot}})

* + - * note field가 존재하거나 exam이 100인 조건에 맞는 학생들에게 total에 가산 점 10점 추가하고 total이 100점이 넘으면 100으로 수정하기 위해 update\_many을 이용.

db.grades.update\_many(

{'$or': [{'note':{'$exists':1}} , {'grades.type':'exam' , 'grades.score':100}]},

{'$inc':{'total':10}} )

db.grades.update\_many({'total':{'$gt':100}}, {'$set' : {'total':100}})

* + - * relative collection 을 생성.

db.create\_collection('relative')

* + - * total과 min, max로 만든 문제에서 제시한 x값과 해당하는 letter를 생성한 수 insert\_one으로 삽입.

db.relative.insert\_one({'sid':sid\_d['sid'], 'total':sid\_d['total'], 'letter':let})

* + - * sid를 ascending으로 정렬하고 letter와 sid와 만 보이게 하기 위해 projection이용하여 result cursor생성.

result = db.relative.find({},{'\_id':0, 'letter':1,'sid':1}).sort([('sid', 1)])

* + 1. 사용 MongoDB함수 및 option
       - update\_one($set) / update\_many($or/$exists/$inc/$gt/$set) / create\_collection / insert\_one / find(projection) / sort(ascending)

1. Pokemon Go

- pokedex.json file import를 위해 import os추가 후 os.system으로 mongo import.

- 실행할 때마다 새로운 Data로 reset을 위한 collection drop function추가.

* 1. problem\_1 : 지우가 잡고 싶은 조건의 포켓몬을 검색하는 함수 구현.
     1. 실행방법 및 실행 code
        + 오보람이 가지고 있는 포켓몬의 문서를 wind\_p에 cursor를 생성한 후 각 문서의 weaknesses field로 공통 약점인 wind\_weak list를 생성.

wind\_p = pokedex.find({'name':{'$in':wind\_pokemon}})

for find\_p in wind\_p:

if wind\_weak == [] : wind\_weak = find\_p['weaknesses']

else : wind\_weak = list(set(wind\_weak)&set(find\_p['weaknesses']))

* + - * wind\_weak의 list에 존재하는 type을 가지고 있고 시간대가 2로 시작하는 때 (20:00~24:00) 출몰하는 포켓몬을 찾아야 함.
      * name을 ascending으로 정렬하고 id, name, spawn\_time, type 값을 보이게 하기 위해 projection이용하여 strong cursor생성

strong = pokedex.find({'type':{'$in':wind\_weak}, 'spawn\_time':{'$regex' : '^2'}},

{'\_id':0, 'id':1, 'name':1, 'spawn\_time':1, 'type':1}).sort([('name',1)])

* + 1. 사용 MongoDB함수 및 option
       - find($in/$regex/projection) / sort(ascending)
  1. problem\_2 : 최종 진화 체가 되기 위한 사탕의 종류와 개수를 출력하는 함수 구현.
     1. 실행방법 및 실행 code
        + 최종진화체를 찾기 위해 next\_evolution field가 없는 문서검색하고 id를 기준으로 ascending하도록 cursor 생성.

final\_pokemons = pokedex.find({'next\_evolution':{'$exists':0}}).sort([('id',1)])

* + - * 각 최종진화체의 name이 next\_evolution.name인 포켓몬이 존재함을 알기 위해 count함수로 확인.

pre\_poke\_cnt = pokedex.find({'next\_evolution.name':pokemon['name']}).count()

포켓몬들의 next\_evolution.name에 각 최종진화체의 name이 들어 있다면 해당 candy의 이름을 저장하고 candy\_count를 누적.

pre\_poke = pokedex.find({'next\_evolution.name':pokemon['name']})

for poke in pre\_poke:

candy = poke['candy']

count += int(poke['candy\_count'])

* + 1. 사용 MongoDB함수 및 option
       - find($exists) / sort(ascending) / embedded document control