C9c- Advanced Mongo

Referencing, importing JSON files, more advanced query

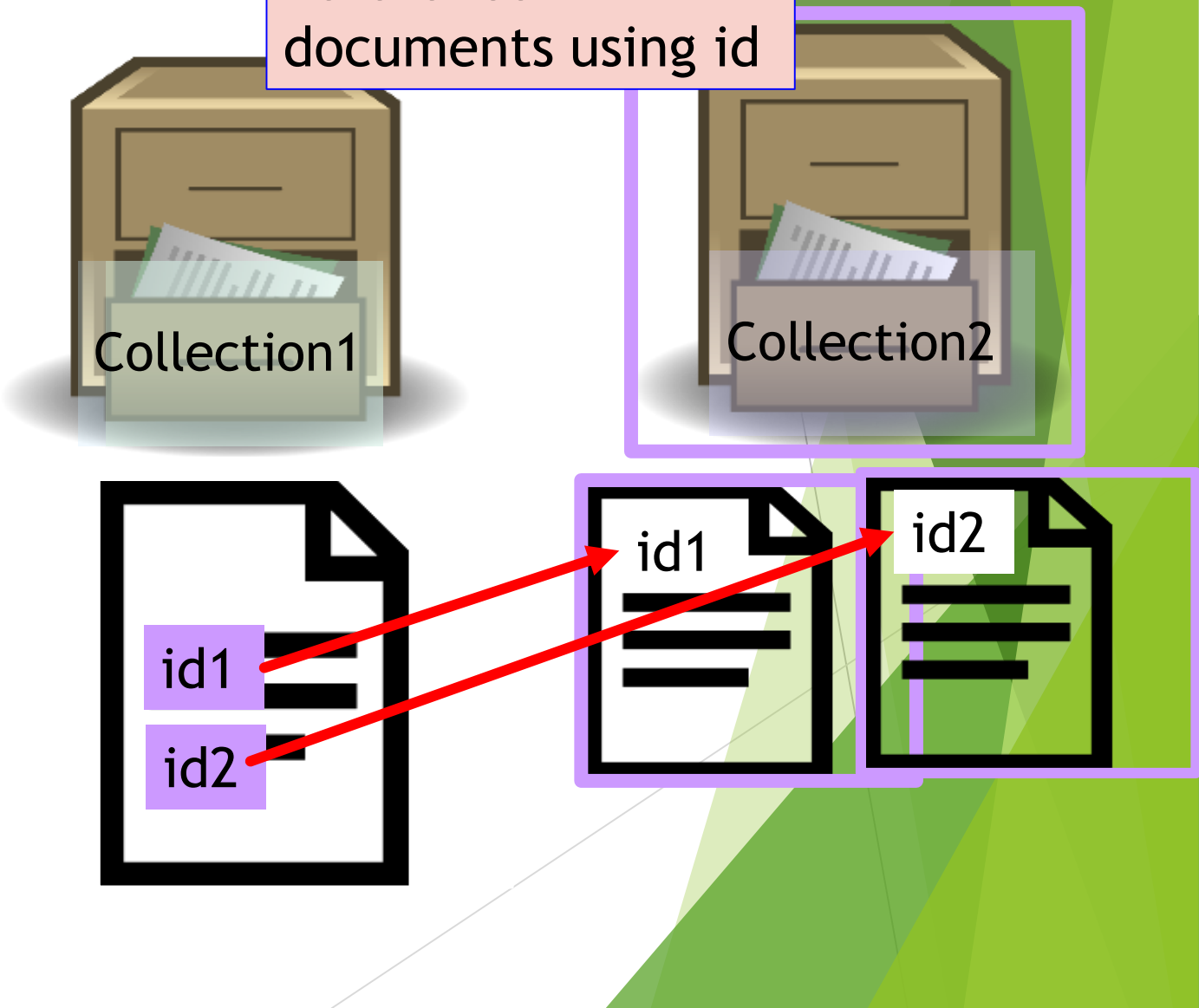
Modelling relationships in Document-based NoSQL

Embedding documents within documents

One-to-one

One-to-many

Reference documents using id



Embedded Relationships

- Documents within Documents

one-to-one

```
{
    "_id": "key1"
    "ID": 1,
    "Name": "John",
    "Age": 25,
    "Address": {
                "street": "Summer Drive",
                "city": "Las Vegas",
                "state": "Nevada",
                "country": "United States"
            }
}
```

one-to-many

```
{
    "_id": "key2"
    "ID": 2,
    "Name": "Mary",
    "Age": 23,
    "CG": "S16",
    "CCA": "dance",
    "tutors": [
                {"name": "Peter",
                  "subject": "Math"},
                {"name": "Lenin",
                  "subject": "Hist"},
                {"name": "Adam",
                  "subject": "Eng"},
              ]
}
```

Document Referenced Relationships

- One-to-many relationship using reference

```
{
    "_id": "T1",
    "name": "Peter",
    "subject": "Math"
}
```

document

```
{
    "_id": "T2",
    "name": "Lenin",
    "subject": "Hist"
}
```

document

```
{
    "_id": "T3",
    "name": "Adam",
    "subject": "Eng"
}
```

document

tutors

Parent document

students

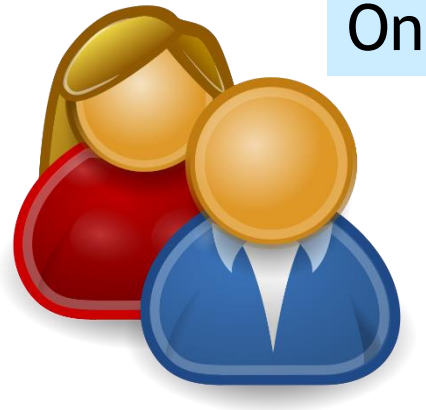
```
{
    "_id": "key2"
    "ID": 2,
    "Name": "Mary",
    "Age": 23,
    "CG": "S16",
    "CCA": "dance",
    "tutor_id": [
                "T1",
                "T2",
                "T3"
                ]
}
```

Child document

document

Social Media Database

Using referencing to establish relationships



Users

One-to-many

Post

One-to-many

Comments

Database: social_media

1. Create a new collection named **users** and insert the following documents:

username: GoodGuy
gender: M
dob: 1/1/1995

username: SweetGirl
gender: F
birthday: 9/9/1996
status: single

To create ISO date objects, use the following:

new Date ("YYYY-MM-DD")

e.g. db.users.insert({ username: "GoodGuy", gender: "M", dob: new Date ("1995-01-01") })

Database: social_media

2. Create a new collection named posts and insert the following documents:

username: GoodGuy
title: Bday bash with the boys!
body: Chilling at the pool
location: MBS
date:1/1/2020

username: GoodGuy
title: StayAtHome
body: Being socially responsible :p
date:1/4/2020

username: SweetGirl
title: New Year New Me
body: Be better each day!
date:1/1/2020

username: SweetGirl
title: Stay Safe
body: Wash your hands!
date:9/3/2020

username: SweetGirl
title: Stay Safe 2
body: Wear a mask!
date:8/4/2020

Database: social_media

3. Create a new collection named comments and insert the following documents:

username: SweetGirl
comment: HBD!
post: <post_obj_id>
*<post_obj_id> is the _id of the post titled Bday Bash with the boys!

username: SweetGirl
comment: Great job!
post: <post_obj_id>
*<post_obj_id> is the _id of the post titled StayAtHome

username: GoodGuy
comment: Happy 2020!
post: <post_obj_id>
*<post_obj_id> is the _id of the post titled New Year New Me

username: GoodGuy
comment: with soap!
post: <post_obj_id>
*<post_obj_id> is the _id of the post titled Stay Safe

Exercise 1

Using the social_media db created, write statement(s) to

- List all posts by GoodGuy
- List all posts that has a location field
- List all posts which are posted after 10/3/2020
- List the post which has the comment “Great Job!”
- Submit screenshots

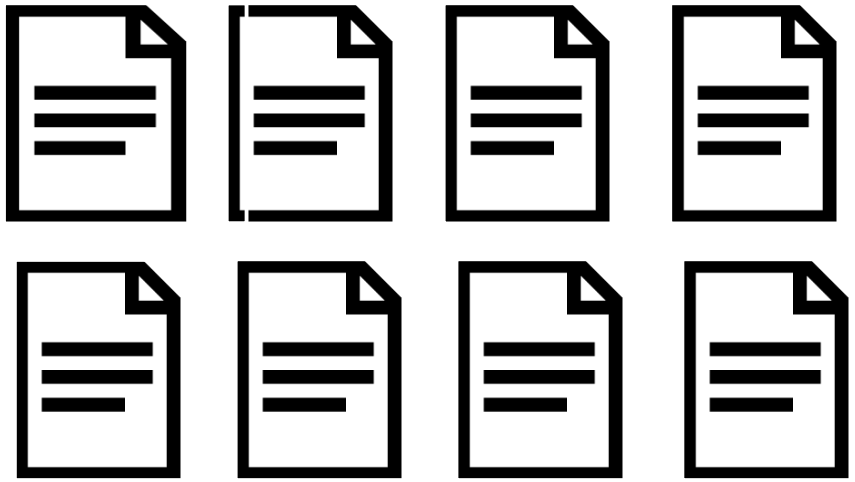
JSON file

- A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format.
- It is primarily used for transmitting data between a web application and a server.
- JSON files are lightweight, text-based, human-readable, and can be edited using a text editor (e.g. Notepad++).

JSON file format

```
{ "name": "Peter", "subject": "Math"}
{ "name": "Johnny", "subject": "Eng"}
{ "name": "Jim", "subject": "PE"}
```

1) JSON objects



```
[
    {
        "name": "Peter",
        "subject": "Math"
    },
    {
        "name": "Johnny",
        "subject": "Eng"
    },
    {
        "name": "Jim",
        "subject": "PE"
    }
]
```

2. Array of JSON objects

Importing JSON file into MongoDB

Download these 2 JSON files:

1. restaurants.json
2. seed.json

For convenience, create a new folder on your desktop named JSON and save these 2 files there.

Importing JSON documents as a collection

1. JSON file with JSON objects

- Open the restaurant.json file. Notice that the file contains 3772 JSON objects.

restaurants.json


```
1 {"address": {"building": "1007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462"}, "borough":
  "Bronx", "cuisine": "Bakery", "grades": [{"date": {"$date": 1393804800000}, "grade": "A", "score": 2}, {"date": {"$date":
  1378857600000}, "grade": "A", "score": 6}, {"date": {"$date": 1358985600000}, "grade": "A", "score": 10}, {"date": {"$date":
  1322006400000}, "grade": "A", "score": 9}, {"date": {"$date": 1299715200000}, "grade": "B", "score": 14}], "name": "Morris Park
  Bake Shop", "restaurant_id": "30075445"}
2 {"address": {"building": "469", "coord": [-73.961704, 40.662942], "street": "Flatbush Avenue", "zipcode": "11225"}, "borough":
  "Brooklyn", "cuisine": "Hamburgers", "grades": [{"date": {"$date": 1419897600000}, "grade": "A", "score": 8}, {"date": {"$date":
  1404172800000}, "grade": "B", "score": 23}, {"date": {"$date": 1367280000000}, "grade": "A", "score": 12}, {"date": {"$date":
  1336435200000}, "grade": "A", "score": 12}], "name": "Wendy'S", "restaurant_id": "30112340"}
3 {"address": {"building": "351", "coord": [-73.98513559999999, 40.7676919], "street": "West   57 Street", "zipcode": "10019"},
  "borough": "Manhattan", "cuisine": "Irish", "grades": [{"date": {"$date": 1409961600000}, "grade": "A", "score": 2}, {"date": {
  "$date": 1374451200000}, "grade": "A", "score": 11}, {"date": {"$date": 1343692800000}, "grade": "A", "score": 12}, {"date": {
  "$date": 1325116800000}, "grade": "A", "score": 12}], "name": "Dj Reynolds Pub And Restaurant", "restaurant_id": "30191841"}
4 {"address": {"building": "2780", "coord": [-73.98241999999999, 40.579505], "street": "Stillwell Avenue", "zipcode": "11224"},
  "borough": "Brooklyn", "cuisine": "American ", "grades": [{"date": {"$date": 1402358400000}, "grade": "A", "score": 5}, {"date": {
  "$date": 1370390400000}, "grade": "A", "score": 7}, {"date": {"$date": 1334275200000}, "grade": "A", "score": 12}, {"date": {"$date"
  : 1318377600000}, "grade": "A", "score": 12}], "name": "Riviera Caterer", "restaurant_id": "40356018"}
5 {"address": {"building": "97-22", "coord": [-73.8601152, 40.7311739], "street": "63 Road", "zipcode": "11374"}, "borough": "Queens",
   "cuisine": "Jewish/Kosher", "grades": [{"date": {"$date": 1416787200000}, "grade": "Z", "score": 20}, {"date": {"$date":
  1358380800000}, "grade": "A", "score": 13}, {"date": {"$date": 1343865600000}, "grade": "A", "score": 13}, {"date": {"$date":
  1323907200000}, "grade": "B", "score": 25}], "name": "Tov Kosher Kitchen", "restaurant_id": "40356068"}
6 {"address": {"building": "8825", "coord": [-73.8803827, 40.7643124], "street": "Astoria Boulevard", "zipcode": "11369"}, "borough":
  "Queens", "cuisine": "American ", "grades": [{"date": {"$date": 1416009600000}, "grade": "Z", "score": 38}, {"date": {"$date":
  1398988800000}, "grade": "A", "score": 10}, {"date": {"$date": 1362182400000}, "grade": "A", "score": 7}, {"date": {"$date":
  1328832000000}, "grade": "A", "score": 13}], "name": "Brunos On The Boulevard", "restaurant_id": "40356151"}
7 {"address": {"building": "2206", "coord": [-74.1377286, 40.6119572], "street": "Victory Boulevard", "zipcode": "10314"}, "borough":
  "Staten Island", "cuisine": "Jewish/Kosher", "grades": [{"date": {"$date": 1412553600000}, "grade": "A", "score": 9}, {"date": {
  "$date": 1400544000000}, "grade": "A", "score": 12}, {"date": {"$date": 1365033600000}, "grade": "A", "score": 12}, {"date": {
  "$date": 1327363200000}, "grade": "A", "score": 9}], "name": "Kosher Island", "restaurant_id": "40356442"}
8 {"address": {"building": "7114", "coord": [-73.9068506, 40.6199034], "street": "Avenue U", "zipcode": "11234"}, "borough":
  "Brooklyn", "cuisine": "Delicatessen", "grades": [{"date": {"$date": 1401321600000}, "grade": "A", "score": 10}, {"date": {"$date":
  1389657600000}, "grade": "A", "score": 10}, {"date": {"$date": 1375488000000}, "grade": "A", "score": 8}, {"date": {"$date":
  1342569600000}, "grade": "A", "score": 10}, {"date": {"$date": 1331251200000}, "grade": "A", "score": 13}, {"date": {"$date":
  1318550400000}, "grade": "A", "score": 9}], "name": "Wilken'S Fine Food", "restaurant_id": "40356483"}
```

Importing JSON documents as a collection

2. Launch Command Prompt

a. Launch Command Prompt by typing cmd after clicking windows icon

```
Command Prompt
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\S90[redacted]C>
```

b. Change directory to where the MongoDB is installed by typing:

- cd C:\Program Files\MongoDB\Server\3.4\bin

```
C:\Users\S90[redacted]C>cd C:\Program Files\MongoDB\Server\3.4\bin

C:\Program Files\MongoDB\Server\3.4\bin>
```

Importing JSON documents as a collection

3. Import a collection of documents from JSON file

- To import the JSON objects into MongoDB, type the following in your command prompt:

mongoimport -d restaurant -c restaurants --file C:\Users\Desktop\JSON\restaurants.json

```
C:\Program Files\MongoDB\Server\3.4\bin>mongoimport -d restaurant -c restaurants --file C:\Users\S90[redacted]C\Desktop\JSON\restaurants.json
2020-04-21T09:05:39.297+0800    connected to: localhost
2020-04-21T09:05:39.559+0800    imported 3772 documents
```

mongoimport: command to import files to mongoDB
-d restaurant: command to connect to db named restaurant, or create one if it's not there
-c restaurants: command to use collection named restaurants, or create one if it's not there
--file C:\Users\Desktop\JSON\restaurants.json: path/directory of the JSON file.
**Note that there should not be any whitespaces in the path

Importing JSON documents as a collection

4. Check that file is imported successfully into MongoDB

In your Mongo client, perform the following:

- Show all available databases
  - show dbs #you should see the restaurant db
- Access restaurant db:
  - use restaurant #you should get 'switched to db restaurant'
- Show collections:
  - Show collections #you should get restaurants
- Display all data:
  - db.restaurants.find().pretty()

```
> db.restaurants.find().pretty()
{
        "_id" : ObjectId("5e9e46e315110507fbadce49"),
        "address" : {
                "building" : "7114",
                "coord" : [
                        -73.9068506,
                        40.6199034
                ],
                "street" : "Avenue U",
                "zipcode" : "11234"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "grades" : [
                {
                        "date" : ISODate("2014-05-29T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2013-08-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2012-07-18T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
```

Importing JSON documents as a collection

5. To narrow display down to certain fields:

- Display only certain fields:

Syntax: db.COLLECTION_NAME.find({ }, {field1:1,field2:1,…})

- db.restaurants.find({ }, {restaurant_id:1, name:1,borough:1,cuisine:1})

```
> db.restaurants.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1}).prett
y()
{
        "_id" : ObjectId("5e9e46e315110507fbadce49"),
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Wilken'S Fine Food",
        "restaurant_id" : "40356483"
}
{
        "_id" : ObjectId("5e9e46e315110507fbadce4a"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Regina Caterers",
        "restaurant_id" : "40356649"
}
{
        "_id" : ObjectId("5e9e46e315110507fbadce4b"),
        "borough" : "Brooklyn",
        "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
        "name" : "Taste The Tropics Ice Cream",
        "restaurant_id" : "40356731"
}
```

Importing JSON documents as a collection

6. To exclude displaying _id field:

- Exclude _id field:

Syntax: db.COLLECTION_NAME.find({ }, {_id:0})

- db.restaurants.find({ }, {_id:0, restaurant_id:1, name:1,borough:1,cuisine:1})

```
> db.restaurants.find({},{"restaurant_id":1,"name":1,"borough":1,"cuisine":1,_id:0}).pretty()
{
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Wilken'S Fine Food",
        "restaurant_id" : "40356483"
}
{
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Regina Caterers",
        "restaurant_id" : "40356649"
}
{
        "borough" : "Brooklyn",
        "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
        "name" : "Taste The Tropics Ice Cream",
        "restaurant_id" : "40356731"
}
```

Exercise 2 (part 1)

Using the restaurant db created, write statement to display

- first 5 restaurants in the borough Bronx
- next 5 restaurants after skipping the first 5 restaurants in borough Bronx
- restaurants in alphabetical order of their names
- restaurants with name of cuisine in alphabetical order, and for that same cuisine, borough should be in descending order
- restaurants that has the street field in their address
- restaurants that belong to borough Bronx and prepared either American or Chinese dish

Exercise 2 (part 2)

Using the restaurant db created, write statement to display

- the restaurant_id, name, borough and cuisine for restaurants which belong to borough Staten Island or Queens or Bronxor or Brooklyn.
- the restaurant_id, name, borough and cuisine, excluding _id, for restaurants which do not belong to borough Staten Island or Queens or Bronxor or Brooklyn.
- Restaurants that achieve a score more than 90
- Restaurants that achieve a score more than 80 but less than 100
- restaurants located in latitude value less than -95.754168
- Restaurants that do not prepare cuisine of ‘American’ and their grade score more than 70

Importing JSON Array documents as a collection

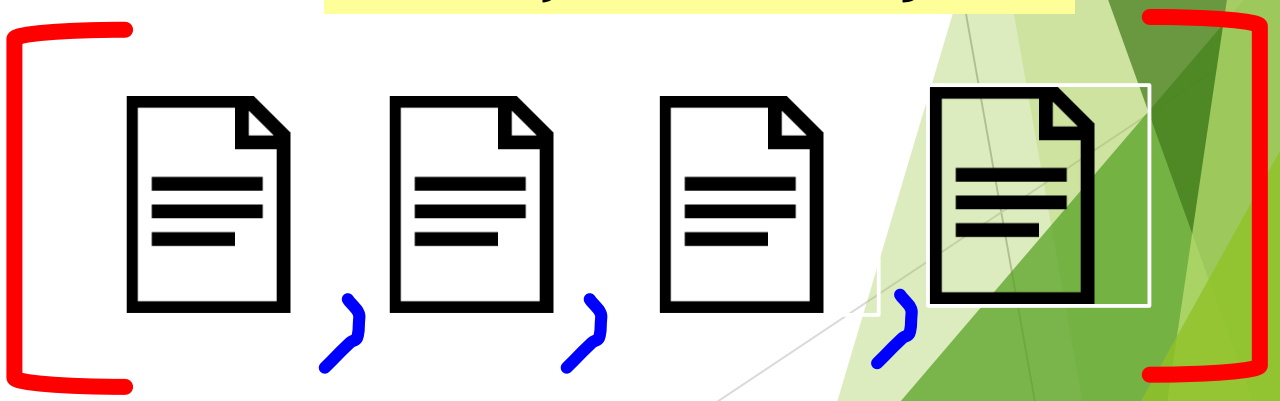
1. JSON file with JSON Array objects

- Open the seed.json file. Notice that the file contains 151 JSON objects contained in an array.

seed.json

```
1  [
2    {
3      "id": "001",
4      "name": "Bulbasaur",
5      "img": "http://img.pokemondb.net/artwork/bulbasaur.jpg",
6      "type": [
7        "Grass",
8        "Poison"
9      ],
10     "stats": {
11       "hp": 45,
12       "attack": "49",
13       "defense": "49",
14       "spattack": "65",
15       "spdefense": "65",
16       "speed": 45
17     },
18     "moves": {
19       "level": [
20         {
21           "learnedat": "",
22           "name": "tackle",
23           "gen": "V"
```

2. Array of JSON objects



Importing JSON Array documents as a collection

2. Launch Command Prompt

a. Launch Command Prompt by typing cmd after clicking windows icon

```
Command Prompt
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\S90[redacted]C>
```

b. Change directory to where the MongoDB is installed by typing:

- cd C:\Program Files\MongoDB\Server\3.4\bin

```
C:\Users\S90[redacted]C>cd C:\Program Files\MongoDB\Server\3.4\bin

C:\Program Files\MongoDB\Server\3.4\bin>
```

Importing JSON Array documents as a collection

3. Import a collection of documents from JSON file

- To import the array of JSON objects into MongoDB, type the following in your command prompt:

mongoimport -d pokemon -c pokemons --file C:\Users\Desktop\JSON\seed.json --jsonArray

```
C:\Program Files\MongoDB\Server\3.4\bin>mongoimport -d pokemon -c pokemons --file C:\Users\S90[redacted]3C\Desktop\JSON\seed.json --jsonArray
2020-04-21T08:57:21.408+0800    connected to: localhost
2020-04-21T08:57:21.591+0800    imported 151 documents
```

mongoimport: command to import files to mongoDB
-d pokemon: command to connect to db named pokemon, or create one if it's not there
-c pokemons: command to use collection named pokemons, or create one if it's not there
--file C:\Users\Desktop\JSON\seed.json: path/directory of the JSON file.
--jsonArray: the type of object contained in the json file.
**Note that there should not be any whitespaces in the path

Importing JSON Array documents as a collection

4. Check that file is imported successfully into MongoDB

In your Mongo client, perform the following:

- Show all available databases
  - show dbs #you should see the restaurant db
- Access restaurant db:
  - use pokemon #you should get ‘switched to db pokemon’
- Show collections:
  - Show collections #you should get pokemons
- Display all data:
  - db.pokemons.find().pretty()

```
> db.pokemons.find().pretty()
{
        "_id" : ObjectId("5e9e44f115110507fbadcda0"),
        "id" : "002",
        "name" : "Ivysaur",
        "img" : "http://img.pokemondb.net/artwork/ivysaur.jpg",
        "type" : [
                "Grass",
                "Poison"
        ],
        "stats" : {
                "hp" : "60",
                "attack" : "62",
                "defense" : "63",
                "spattack" : "80",
                "spdefense" : "80",
                "speed" : "60"
        },
        "moves" : {
                "level" : [
                        {
                                "learnedat" : "",
                                "name" : "tackle",
                                "gen" : "V"
                        },
```

Importing JSON Array documents as a collection

5. To narrow display down to certain fields:

- Display only certain fields:

Syntax: db.COLLECTION_NAME.find({ }, {field1:1,field2:1,…})

- db.pokemons.find({ }, {id:1, name:1,type:1,img:1})

```
> db.pokemons.find({},{id:1,name:1,type:1,img:1}).pretty()
{
        "_id" : ObjectId("5e9e44f115110507fbadcda0"),
        "id" : "002",
        "name" : "Ivysaur",
        "img" : "http://img.pokemondb.net/artwork/ivysaur.jpg",
        "type" : [
                "Grass",
                "Poison"
        ]
}
{
        "_id" : ObjectId("5e9e44f115110507fbadcda1"),
        "id" : "008",
        "name" : "Wartortle",
        "img" : "http://img.pokemondb.net/artwork/wartortle.jpg",
        "type" : [
                "Water"
        ]
}
{
        "_id" : ObjectId("5e9e44f115110507fbadcda2"),
        "id" : "004",
        "name" : "Charmander",
        "img" : "http://img.pokemondb.net/artwork/charmander.jpg",
        "type" : [
                "Fire"
        ]
}
```

Importing JSON Array documents as a collection

6. To exclude displaying _id field:

- Exclude _id field:

Syntax: db.COLLECTION_NAME.find({ }, {_id:0})

- db.pokemons.find({ }, {_id:0, id:1, name:1,type:1,img:1})

```
> db.pokemons.find({},{_id:0,id:1,name:1,type:1,img:1}).pretty()
{
        "id" : "002",
        "name" : "Ivysaur",
        "img" : "http://img.pokemondb.net/artwork/ivysaur.jpg",
        "type" : [
                "Grass",
                "Poison"
        ]
}
{
        "id" : "008",
        "name" : "Wartortle",
        "img" : "http://img.pokemondb.net/artwork/wartortle.jpg",
        "type" : [
                "Water"
        ]
}
{
        "id" : "004",
        "name" : "Charmander",
        "img" : "http://img.pokemondb.net/artwork/charmander.jpg",
        "type" : [
                "Fire"
        ]
}
```

Exercise 3 (part 1)

Using the pokemon db created, write statement to

- Display only the names of all the pokemons
- Find Pokemon with the name “Mew”
- Count Pokemons who are 87.5% male
- Count Pokemons who have ice damage that is “2”
- Display Pokemons who have ice damage that is “2” AND 12.5% female in alphabetical order of their name
- Count Pokemons who have speed: “60” OR type: “Grass”
- Count Pokemons who have speed: “60” AND type: “Grass”

Exercise 3 (part 2)

Using the pokemon db created, write statement to

- Display only id, name, type and stats of all pokemon
- Select the top 3 Pokemons according to their attack
- Count pokemons who are “Bug” type
- Sort all “Fire” type pokemon according to their hp, displaying first 5
- Count pokemons who have poison damage >0 AND psychic damange >= 1
- Sort Flying type Pokemon in descending order of defense
- Count pokemons with happiness >= 70