Create in MongoDB

Watch video: https://youtu.be/i7-yTCk8oV8?t=3361

grades.json

- In our MongoDB labs, we will work with a sample dataset of grades
- 280 documents with this structure:


```
{
  "_id" : { "$oid" : "50b59cd75bed76f46522c34e" },
  "student_id" : 0,
  "class_id" : 2,
  "scores" : [
    { "type" : "exam", "score" : 57.92947112575566 },
    { "type" : "quiz", "score" : 21.24542588206755 },
    { "type" : "homework", "score" : 68.19567810587429 },
    { "type" : "homework", "score" : 67.95019716560351 },
    { "type" : "homework", "score" : 18.81037253352722 }
  ]
}
```

Importing grades.json

- Use mongoimport utility (in system command prompt)
  - -d grades: creates grades database
  - -c grades: creates grades collection
  - --drop: drops the collection if it already existed
  - grades.json: name of the file to import (note that this file is in the mongo bin folder; specify full path if in a different location

```
mongoimport -d grades -c grades --drop grades.json
```

db.collection.insert()

- The insert() method has the following syntax:

```
db.products.insert( { _id: 10, item: "box", qty: 20 } )
```

Insert without specifying an _id field

- In the following example, the document passed to the insert() method does not contain the _id field:

```
db.products.insert( { item: "card", qty: 15 } )
```

- During the insert, mongod will create the _id field and assign it a unique ObjectId value, as verified by the inserted document:

```
{ "_id" : ObjectId("5063114bd386d8fadbd6b004"), "item" : "card", "qty" : 15 }
```

Multiple inserts

- The following example performs a bulk insert of three documents by passing an array of documents to the insert() method.

```
db.products.insert(
   [
     { _id: 11, item: "pencil", qty: 50, type: "no.2" },
     { item: "pen", qty: 20 },
     { item: "eraser", qty: 25 }
   ]
)
```

Ordered vs. Unordered inserts

- By default, MongoDB performs an ordered insert. With ordered inserts, if an error occurs during an insert of one of the documents, MongoDB returns an error without processing the remaining documents in the array.

- With unordered inserts, if an error occurs during an insert of one of the documents, MongoDB continues to insert the remaining documents in the array.

Ordered vs. Unordered inserts

- The following example performs an unordered insert of three documents.

```
try {
   db.products.insertMany( [
      { _id: 13, item: "stamps", qty: 125 },
      { _id: 13, item: "tape", qty: 20},
      { _id: 14, item: "bubble wrap", qty: 30}
   ], { ordered: false } );
} catch (e) {
   print (e);
}
```

insert(), insertOne() and insertMany()

- As you have seen, you can use the insert() statement to insert one document and to insert multiple documents.

- However, the insert() method is deprecated and no longer compatible with some MongoDB drivers and external applications, so insertOne() and insertMany() are used instead.

- For us, working in the console, any of the options will work.

insertOne()

- The insertOne() method has the following syntax:

```
db.products.insertOne( { item: "card", qty: 15 } )
```

insertMany()

- The insertMany() method has the following syntax:

```
try {
   db.products.insertMany( [
      { item: "card", qty: 15 },
      { item: "envelope", qty: 20 },
      { item: "stamps" , qty: 30 }
   ] );
} catch (e) {
   print (e);
}
```