Practice 5: MongoDB Basic Queries

Big Data System Design

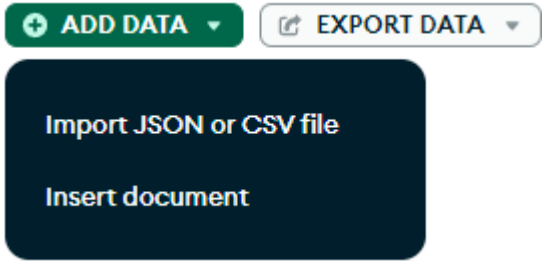
PRACTICE

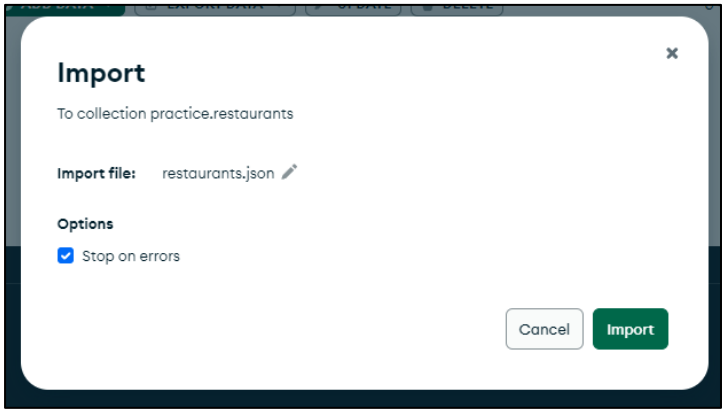
Import dataset

- Creating a new collection
  - use [DATABASE_NAME]
  - db.create_collection(name)

```
>_MONGOSH
> use practice
< switched to db practice
> db.createCollection("restaurants")
< { ok: 1 }
> show collections
< cappedCollection
  inventory
  restaurants
  students
practice>
```

- Import the practice dataset, 'restaurants.json'
  - Download 'restaurants.json' from e-Campus
  - Use import function in MongoDB compass





Import dataset

MongoDB Compass - localhost:27017/practice.restaurants

Connect Edit View Collection Help

localhost:27017

My Queries
Performance
Databases

Search

- admin
- config
- local
- practice
  - cappedCollection
  - inventory
  - restaurants
  - students

My Queries | restaurants

practice > restaurants

Documents 3.8K | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

Explain | Reset | Find | Options

ADD DATA | EXPORT DATA | UPDATE | DELETE

1 – 20 of 3772

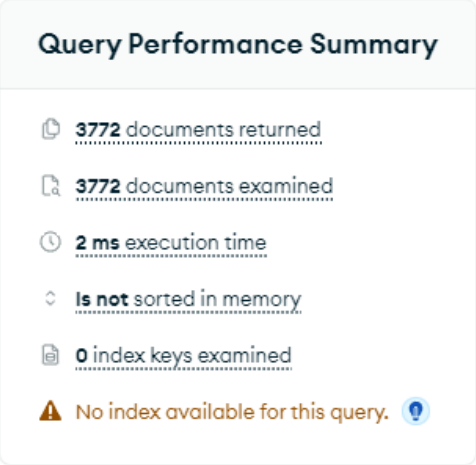
```
_id: ObjectId('660a367696d09f7687d0accf')
address : Object
borough : "Bronx"
cuisine : "Bakery"
grades : Array (5)
name : "Morris Park Bake Shop"
restaurant_id : "30075445"

_id: ObjectId('660a367696d09f7687d0acd0')
address : Object
borough : "Brooklyn"
cuisine : "Hamburgers"
grades : Array (4)
name : "Wendy'S"
restaurant_id : "30112340"

_id: ObjectId('660a367696d09f7687d0acd1')
address : Object
borough : "Manhattan"
cuisine : "Irish"
grades : Array (4)
name : "Dj Reynolds Pub And Restaurant"
restaurant_id : "30191841"

Id('660a367696d09f7687d0acd2')
```

Import completed.
3772 documents imported.

Basic queries

- find() function
  - To query documents from the target collection
  - Syntax
    - db.COLLECTION_NAME.find(query, projection, options)

  - Practice 1: select all documents in a collection
    - db.COLLECTION_NAME.find()



Basic queries

- Projection
  - Determine which fields are returned in the matching documents
    - Syntax
      - {<field 1>: <value>, <field 2>: <value>, …}
      - 1 or true to include the field in the return documents
      - 0 or false to exclude the field
  - Practice 2: write a query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection

Basic queries

- Projection
  - Practice 2 (cont'd): Shell
    - db.restaurants.find({}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})

```
>_MONGOSH
> db.restaurants.find({}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
< {
    _id: ObjectId('660a367696d09f7687d0accf'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd1'),
    borough: 'Manhattan',
    cuisine: 'Irish',
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd2'),
    borough: 'Brooklyn',
```

Basic queries

- Projection
  - Practice 2 (cont'd): Compass

Documents 3.8K | Aggregations | Schema | Indexes 1 | Validation

{} Generate query Explain Reset Find Options

Project {restaurant_id: 1, name: 1, borough: 1, cuisine: 1}

Sort { field: -1 } or [['field', -1]] MaxTimeMS 60000

Collation { locale: 'simple' } Skip 0 Limit 0

EXPORT DATA 1 – 20 of 3772

```
_id: ObjectId('660a367696d09f7687d0accf')
borough : "Bronx"
cuisine : "Bakery"
name : "Morris Park Bake Shop"
restaurant_id : "30075445"
```

```
_id: ObjectId('660a367696d09f7687d0acd0')
borough : "Brooklyn"
cuisine : "Hamburgers"
name : "Wendy'S"
restaurant_id : "30112340"
```

```
_id: ObjectId('660a367696d09f7687d0acd1')
borough : "Manhattan"
cuisine : "Irish"
name : "Dj Reynolds Pub And Restaurant"
```

Basic queries

- Projection
  - Task 1: write a query to display the fields restaurant_id, name, borough, and zip code, but exclude the field _id for all documents in the collection

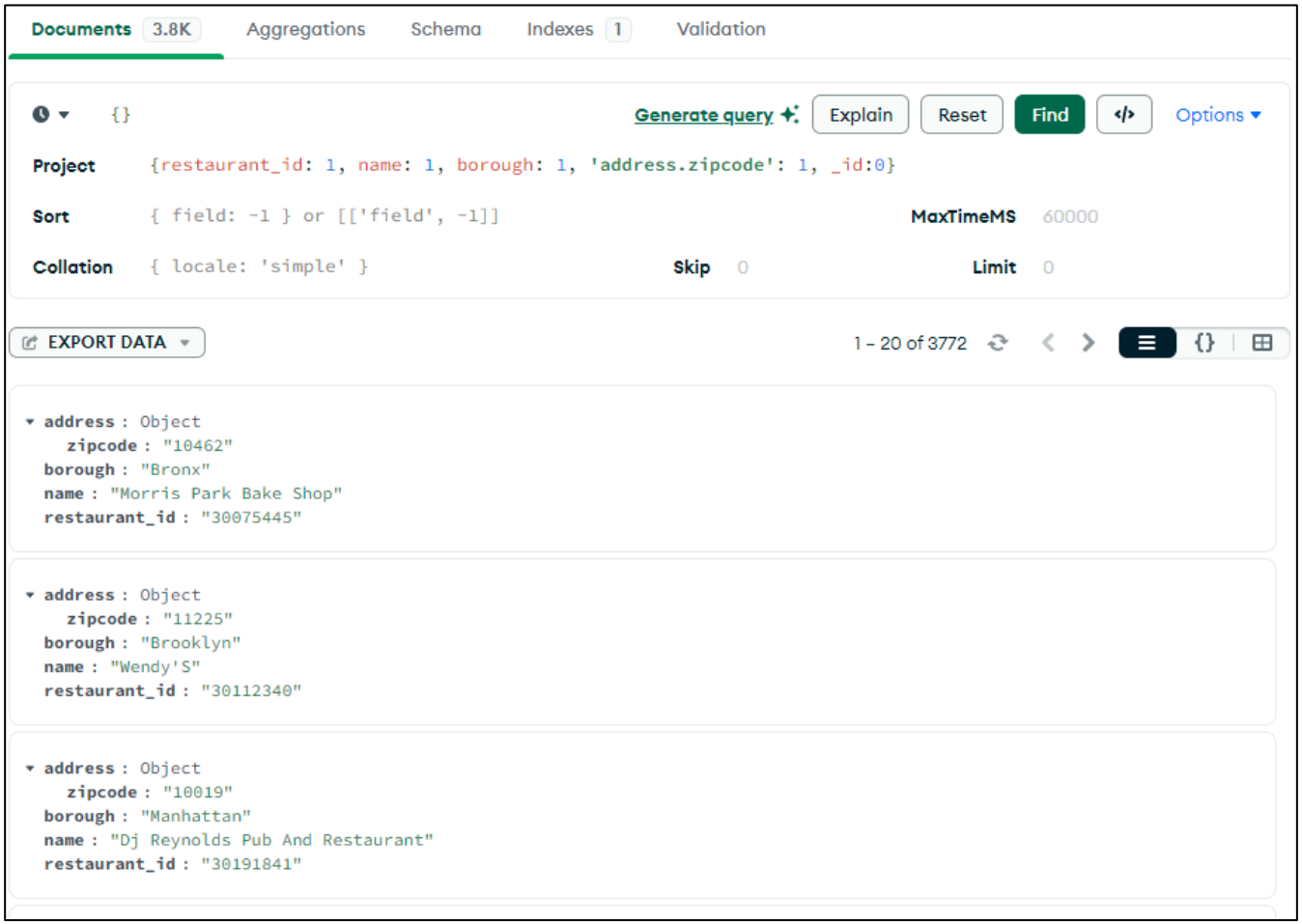
Basic queries

- Projection
  - Task 1 (cont'd): Shell
    - db.restaurants.find({}, {_id: 0, restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1})

```
>_MONGOSH
> db.restaurants.find({}, {_id: 0, restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1})
< {
    address: {
      zipcode: '10462'
    },
    borough: 'Bronx',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    address: {
      zipcode: '11225'
    },
    borough: 'Brooklyn',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    address: {
      zipcode: '10019'
    },
    borough: 'Manhattan'
```

Basic queries

- Projection
  - Task 1 (cont'd): Compass



Basic queries

- Comparison operator: $eq
  - Match all values that are equal to the specific values
  - Expression
    - {<field>: {$eq: <value>}}
    - {<field>: <value>}

  - Practice 3
    - Find the restaurant_id, name, borough, and cuisine for those restaurants which belong to the borough Queens

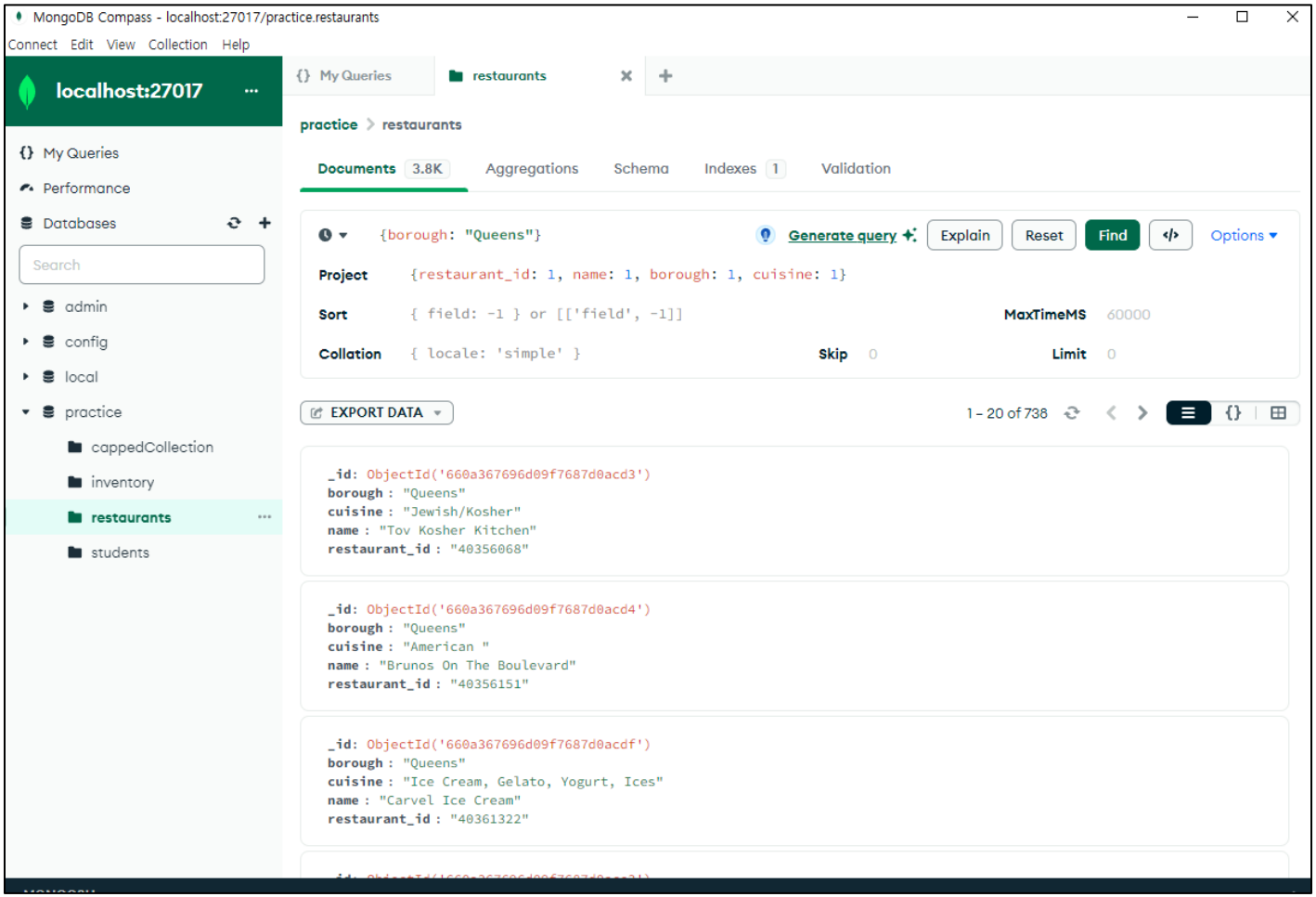
Basic queries

- Comparison operator: $eq
  - Practice 3 (cont'd): Shell
    - db.restaurants.find({borough: "Queens"}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})

```
>_MONGOSH
> db.restaurants.find({borough: "Queens"}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
< {
    _id: ObjectId('660a367696d09f7687d0acd3'),
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    name: 'Tov Kosher Kitchen',
    restaurant_id: '40356068'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd4'),
    borough: 'Queens',
    cuisine: 'American ',
    name: 'Brunos On The Boulevard',
    restaurant_id: '40356151'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acdf'),
    borough: 'Queens',
    cuisine: 'Ice Cream, Gelato, Yogurt, Ices',
    name: 'Carvel Ice Cream',
    restaurant_id: '40361322'
  }
  {
    _id: ObjectId('660a367696d09f7687d0ace3'),
    borough: 'Queens',
    cuisine: 'Delicatessen',
```

Basic queries

- Comparison operator: $eq
  - Practice 3 (cont'd): Compass



Basic queries

- Comparison operator: $ne
  - Match all values that are not equal to the specific values
  - Expression
    - {<field>: {$ne: <value>}}
  - Practice 4
    - Find the restaurant_id, name, borough, and cuisine for those restaurants which are not belong to the borough Queens

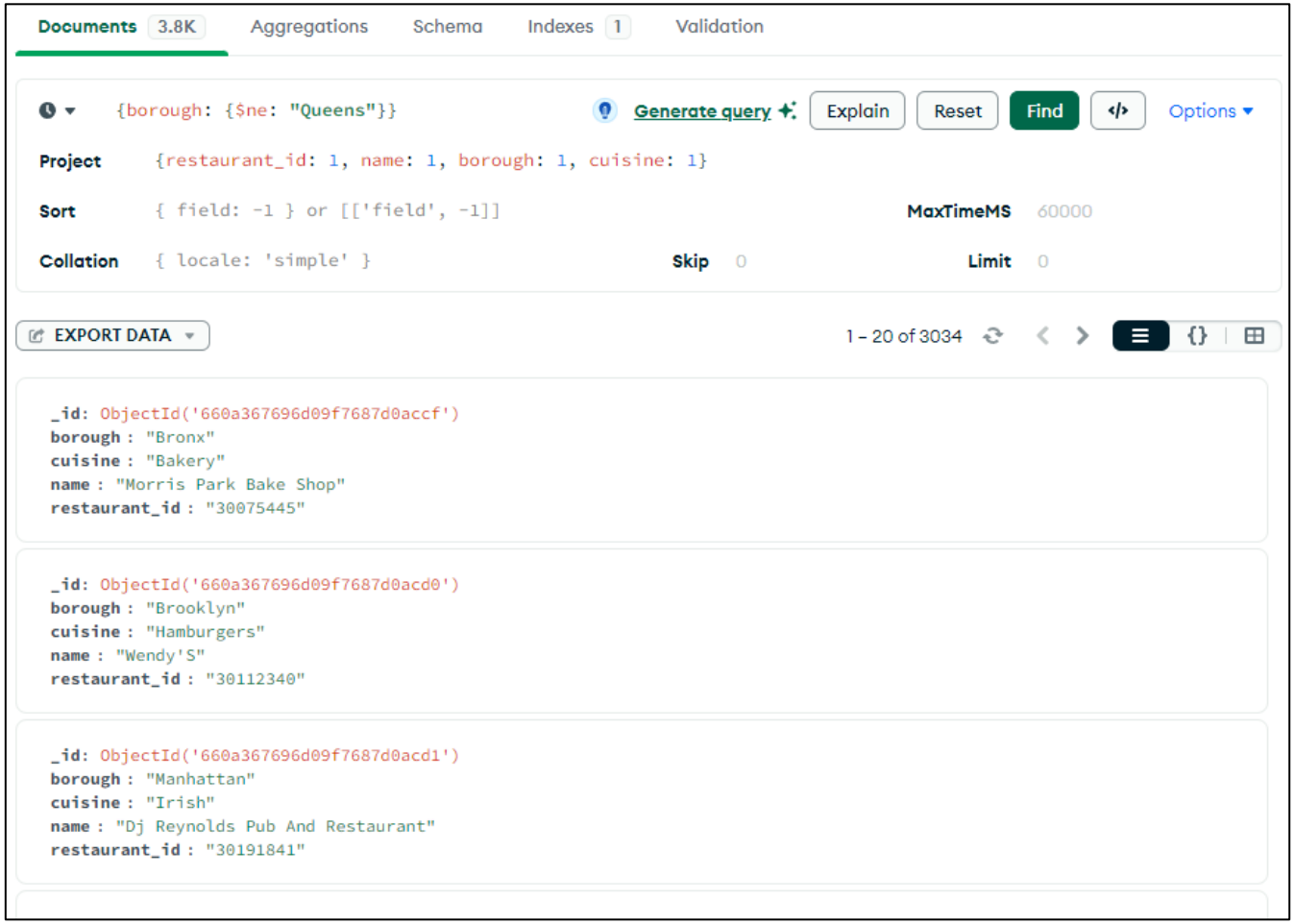
Basic queries

- Comparison operator: $ne
  - Practice 4 (cont'd): Shell

```
>_MONGOSH

> db.restaurants.find({borough: {$ne: "Queens"}}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
< {
    _id: ObjectId('660a367696d09f7687d0accf'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd1'),
    borough: 'Manhattan',
    cuisine: 'Irish',
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd2'),
    borough: 'Brooklyn',
    cuisine: 'American ',
```

Basic queries

- Comparison operator: $ne
  - Practice 4 (cont'd): Compass



Basic queries

- Comparison operator: $ne
  - Task 2
    - Write a query to find the restaurant_id, name, borough, and street for those restaurants which are not belonging to the Bronx

Basic queries

- Comparison operator: $ne
  - Task 2 (cont'd): Shell

```
>_MONGOSH
> db.restaurants.find({borough: {$ne: "Bronx"}},{restaurant_id: 1, name: 1, borough: 1, "address.street": 1})
< {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    address: {
      street: 'Flatbush Avenue'
    },
    borough: 'Brooklyn',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd1'),
    address: {
      street: 'West   57 Street'
    },
    borough: 'Manhattan',
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd2'),
    address: {
      street: 'Stillwell Avenue'
    },
    borough: 'Brooklyn',
    name: 'Riviera Caterer'
```

Basic queries

- Comparison operator: $ne
  - Task 2 (cont'd): Compass

Documents 3.8K | Aggregations | Schema | Indexes 1 | Validation

```
{borough: {$ne: "Bronx"}}
```

Generate query | Explain | Reset | Find | Options

Project `{restaurant_id: 1, name: 1, borough: 1, 'address.street': 1}`

Sort `{ field: -1 } or [['field', -1]]` MaxTimeMS 60000

Collation `{ locale: 'simple' }` Skip 0 Limit 0

EXPORT DATA 1 – 20 of 3463

```
_id: ObjectId('660a367696d09f7687d0acd0')
address : Object
  street : "Flatbush Avenue"
borough : "Brooklyn"
name : "Wendy'S"
restaurant_id : "30112340"

_id: ObjectId('660a367696d09f7687d0acd1')
address : Object
  street : "West   57 Street"
borough : "Manhattan"
name : "Dj Reynolds Pub And Restaurant"
restaurant_id : "30191841"

_id: ObjectId('660a367696d09f7687d0acd2')
address : Object
  street : "Stillwell Avenue"
borough : "Brooklyn"
name : "Riviera Caterer"
```

Basic queries

- Comparison operator: $gt, $gte, $lt, $lte
  - Select documents where the value of the field is greater, greater or equal, less, less or equal than the specific value
  - Expression
    - {<field>: {[$gt | $gte | $lt | $let]: <value>}}

  - Practice 5
    - Write a query to find all restaurants where the first element of 'grades' has a 'score' of 3 or more

Basic queries

- Comparison operator: $gt, $gte, $lt, $lte
  - Practice 5 (cont'd): Shell
    - db.restaurants.find({"grades.0.score": {$gte: 3}})

```
>_MONGOSH
> db.restaurants.find({"grades.0.score": {$gte: 3}})
< {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    address: {
      building: '469',
      coord: [
        -73.961704,
        40.662942
      ],
      street: 'Flatbush Avenue',
      zipcode: '11225'
    },
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    grades: [
      {
        date: 2014-12-30T00:00:00.000Z,
        grade: 'A',
        score: 8
      },
      {
        date: 2014-07-01T00:00:00.000Z,
        grade: 'B',
        score: 23
      },
      {
```

Basic queries

- Logical operator: $and, $or
  - Performs a logical AND, OR operation on an array of two or more expressions
  - Syntax
    - {$and: [{<expression 1>}, {<expression 2>}, …, {<expression N>}]}
  - Task 3
    - Find the restaurants that do not prepare any cuisine of 'American ' and their grade score more than 70

Basic queries

- Logical operator: $and, $or
  - Task 3 (cont'd)

```
>_MONGOSH
> db.restaurants.find({"grades.score": {$gt: 70}, cuisine: {$ne: "American "}})
< {
    _id: ObjectId('660a367696d09f7687d0aece'),
    address: {
      building: '345',
      coord: [
        -73.9864626,
        40.7266739
      ],
      street: 'East 6 Street',
      zipcode: '10003'
    },
    borough: 'Manhattan',
    cuisine: 'Indian',
    grades: [
      {
        date: 2014-09-15T00:00:00.000Z,
        grade: 'A',
        score: 5
      },
      {
        date: 2014-01-14T00:00:00.000Z,
        grade: 'A',
        score: 8
      },
```

Basic queries

- Logical operator: $and, $or
  - Task 3 (cont'd)

Query Performance Summary

- **10** documents returned
- **3772** documents examined
- **5 ms** execution time
- **Is not** sorted in memory
- **0** index keys examined
- No index available for this query.

Query Performance Summary

- **5** documents returned
- **3772** documents examined
- **4 ms** execution time
- **Is not** sorted in memory
- **0** index keys examined
- No index available for this query.

Query Performance Summary

- **4** documents returned
- **3772** documents examined
- **4 ms** execution time
- **Is not** sorted in memory
- **0** index keys examined
- No index available for this query.

Basic queries

- Logical operator: $and, $or
  - Task 4
    - Find the restaurant_id, name, borough, and cuisine for those restaurants which belong to the borough staten Island or Queens or Bronx or Brooklyn

Basic queries

- Logical operator: $and, $or
  - Task 4 (cont'd)
    - db.restaurants.find({$or: [{borough: "Staten Island"},
      {borough: "Queens"}, {borough: "Bronx"},
      {borough: "Brooklyn"}]},
      {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})

```
>_MONGOSH
> db.restaurants.find({$or: [{borough: "Staten Island"}, {borough: "Queens"}, {borough: "Bronx"}, {borough: "Brooklyn"}]}, {restaurant_id: 1, name: 1,
< {
    _id: ObjectId('660a367696d09f7687d0accf'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
```

Basic queries

- Logical operator: $and, $or
  - Task 4 (cont'd)
    - db.restaurants.find({borough: {$in: ["Staten Island", "Queens", "Bronx", "Brooklyn"]}}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})

```
>_MONGOSH
> db.restaurants.find({borough: {$in: ["Staten Island", "Queens", "Bronx", "Brooklyn"]}}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})
< {
    _id: ObjectId('660a367696d09f7687d0accf'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd0'),
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acd2'),
```

Basic queries

- Task 5
  - Find the restaurant_id, name, borough, and cuisine for those restaurants which are not belonging to the borough “Staten Island” or “Queens” or “Bronx” or “Brooklyn”

Basic queries

- Task 5 (cont'd)
  - db.restaurants.find({$nor: [{borough: "Staten Island"}, {borough: "Queens"}, {borough: "Bronx"}, {borough: "Brooklyn"}]}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1})

```
>_MONGOSH
> db.restaurants.find({$nor: [{borough: "Staten Island"}, {borough: "Queens"}, {borough: "Bronx"}, {borough: "Brooklyn"}]}, {restaurant_id: 1, name: 1,
< {
    _id: ObjectId('660a367696d09f7687d0acd1'),
    borough: 'Manhattan',
    cuisine: 'Irish',
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  }
  {
    _id: ObjectId('660a367696d09f7687d0acdc'),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480'
  }
  {
    _id: ObjectId('660a367696d09f7687d0ace1'),
```

Basic queries

- Task 6
  - Find the restaurants which do not prepare any cuisine of “American ” and achieved a grade point “A” not belongs to the borough Brooklyn

Basic queries

- Task 6 (cont'd)
  - db.restaurants.find({cuisine: {$not: {$regex: "American"}},
    "grades.grade": "A", borough: {$ne: "Brooklyn"}})

```
>_MONGOSH
> db.restaurants.find({cuisine: {$not: {$regex: "American"}}, "grades.grade": "A", borough: {$ne: "Brooklyn"}})
< {
    _id: ObjectId('660a367696d09f7687d0accf'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
      }
```

Basic queries

- Final task
  - Find the restaurant_id, name, borough, and cuisine, but exclude _id field for hamburger restaurants which achieved a grade “A” not belongs to the borough Manhattan or Queens or Staten Island or Bronx (using $nin operator)

Basic queries

- Final task (cont'd)
  - db.restaurants.find({cuisine: {$regex: "Hamburgers"}, "grades.grade": "A", borough: {$nin: ["Manhattan", "Queens", "Staten Island", "Bronx"]}}, {restaurant_id: 1, name: 1, borough: 1, cuisine: 1, _id: 0})

```
>_MONGOSH
> db.restaurants.find({cuisine: {$regex: "Hamburger"}, "grades.grade": "A", borough: {$nin: ["Manhattan", "Queens", "Staten Island", "Bronx"]}}, {resta
< {
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Wendy'S",
    restaurant_id: '30112340'
  }
  {
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: 'White Castle',
    restaurant_id: '40362344'
  }
  {
    borough: 'Brooklyn',
    cuisine: 'Hamburgers',
    name: "Mcdonald'S",
    restaurant_id: '40367790'
```

Report for Lecture 5

- Create a MongoDB collection that contains at least 10 documents (also include embedded documents)
  - Optionally, you can also find a dataset and use it

- Create various queries
  - Use comparison operators
  - Use logical operators
  - Use dot notation
  - Use a combination of comparison and logical operators and dot notation (more than two conditions)
  - Use a combination of comparison and logical operators and dot notation (more than three conditions)

- Deadline: April 9, 2024 23:59:59

Questions?

SEE YOU NEXT TIME!