

**LAPORAN UJIAN AKHIR
SEMESTER GENAP
MATKUL BASIS DATA LANJUT**



**OLEH
ANGGER NUR AMIN (21201087)**

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI DAN DESAIN
INSTITUT TEKNOLOGI DAN BISNIS ASIA MALANG
2022/2023**

Instruksi :

1. Kerjakan soal-soal dibawah ini, ikuti petunjuk/ccontoh yang ada.
2. Minimal 20 soal terjawab (boleh lebih), semakin banyak jawaban benar, semakin baik
3. Soal contoh tidak bisa dipilih Kembali.
4. Kumpulkan dalam bentuk (nama_file.pdf)

Contoh :

1. Impor file (.json) ke dalam MongoDB?

Jawab:

Query: mongoimport --db restoran --collection data --file c:\data\db\dbResto.json

Hasil:

```
C:\WINDOWS\system32>mongoimport --db restoran --collection data --file c:\data\db\dbResto.json
2023-07-13T10:43:12.458+0700 connected to: mongod://localhost/
2023-07-13T10:43:15.409+0700 [#####.....] restoran.data 594KB/2.13MB (27.3%)
2023-07-13T10:43:18.355+0700 [#####.....] restoran.data 1.71MB/2.13MB (80.2%)
2023-07-13T10:43:19.567+0700 [#####.....] restoran.data 2.13MB/2.13MB (100.0%)
2023-07-13T10:43:19.701+0700 3772 document(s) imported successfully. 0 document(s) failed to import.
```

Note: sesuaikan dengan nama database dan koleksi yang ingin digunakan, perhatikan juga posisi file .json berada.

2. Hitung rata-rata rating restoran di setiap kota?

Jawab:

Query: db.data.aggregate([{\$group: {_id: "\$borough", averageRating:{\$avg:{\$avg: "\$grades.score"}}}}])

Hasil:

```
> db.data.aggregate([ { $group: { _id: "$borough", averageRating: { $avg: { $avg: "$grades.score" } } } })
{ "_id" : "Queens", "averageRating" : 11.541666666666666 }
{ "_id" : "Bronx", "averageRating" : 17.2 }
{ "_id" : "Brooklyn", "averageRating" : 11.65 }
{ "_id" : "Staten Island", "averageRating" : 12.666666666666668 }
{ "_id" : "Manhattan", "averageRating" : 12.581462585034014 }
>
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. **.aggregate** operator untuk menghitung rata-rata
- c. **\$group** berfungsi untuk mengelompokkan data berdasarkan bidang tertentu, dalam hal ini dikelompokkan pada field borough.
- d. **_id: "\$borough"** pengelompokan data berdasarkan field borough, hasilnya setiap grup akan memiliki ID yang berbeda sesuai dengan kota.
- e. **averageRating: { \$avg: { \$avg : "\$grades.score" } }** berfungsi untuk menghitung rata-rata dari bidang rating di setiap grup, dan **\$avg** akan menghitung rata-rata dari semua nilai rating yang ada dalam grup tersebut.

=====

3. Urutkan restoran berdasarkan rating secara menurun?

JAWAB DISINI SEPerti CONTOH DIATAS

4. Tampilkan restoran yang berlokasi di kota tertentu?

Jawab:

Query: db.data.find({

```

borough: {
  $eq: "Bronx",
},
});

```

Hasil:

```

restoran> db.data.find({
...   borough: {
...     $eq: "Bronx",
...   },
... });
[
  {
    _id: ObjectId("6475511755638321bee73126"),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate("2014-03-03T00:00:00.000Z"),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate("2013-09-11T00:00:00.000Z"),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate("2013-01-24T00:00:00.000Z"),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate("2011-11-23T00:00:00.000Z"),
        grade: 'A',
        score: 9
      },
      {

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
 - b. Menampilkan data yang borough (wilayah) === Bronx pada semua field dengan operator **\$eq** (equal)
5. Menghitung jumlah restoran di setiap kota

Jawab:

Query:

```

db.data.find({
  borough: {

```

```

    $eq: "Brooklyn",
  },
}).count();

```

Hasil:

```

restoran> db.data.find({
...   borough: {
...     $eq: "Brooklyn",
...   },
... }).count();
684

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. Menampilkan jumlah restoran yang berada di Wilayah Brooklyn dengan fungsi **\$eq** (equal)
- c. **\$count()**, berfungsi untuk menghitung / mengambil jumlah data hasil query
6. Tampilkan restoran yang melayani makanan tertentu (misalnya, "Italian" atau "Chinese")?

Jawab:

Query:

```

db.data.find({
  $or: [
    { cuisine: { $in: ["American ", "African"] } },
  ]
})

```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee7315e"),
  address: {
    building: '87-69',
    coord: [ -73.8309503, 40.7001121 ],
    street: 'Lefferts Boulevard',
    zipcode: '11418'
  },
  borough: 'Queens',
  cuisine: 'American ',
  grades: [

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. Akan menampilkan restoran yang melayani makanan "American " atau "African" dengan logical query operator **\$or**
7. Mengubah informasi restoran, seperti alamat atau nomor telepon?

Jawab:

Query:

```

db.data.updateOne(
  {
    _id: ObjectId("6475511755638321bee7313a"),
  },

```

```
{
  $set: {
    borough: "Malang",
  },
}
);
```

Hasil:

```
... }));
[
  {
    _id: ObjectId("6475511755638321bee7313a"),
    address: {
      building: '2780',
      coord: [ -73.98241999999999, 40.579505 ],
      street: 'Stillwell Avenue',
      zipcode: '11224'
    },
    borough: 'Malang',
    cuisine: 'American ',
    grades: [
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
 - b. Update field borough(wilayah) menjadi borough Malang dengan query **\$eq** yang digunakan untuk mencocokkan id ===
ObjectId("6475511755638321bee7313a")
 - c. **\$set**, Berfungsi untuk mengubah nilai field
8. Tampilkan semua restoran yang berlokasi di Manhattan.

Jawab:

Query:

```
db.data.find({
  borough: {
    $eq: "Manhattan",
  },
});
```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee7315d"),
  address: {
    building: '625',
    coord: [ -73.990494, 40.7569545 ],
    street: '8 Avenue',
    zipcode: '10018'
  },
  borough: 'Manhattan',
  cuisine: 'American ',
  grades: [
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.

- b. Menampilkan semua data dengan borough === "Manhattan" menggunakan query \$eq

9. Tampilkan restoran dengan nama "Angelo Of Mulberry St."

Jawab:

Query:

```
db.data.find({
  name: {
    $eq: "Angelo Of Mulberry St.",
  },
});
```

Hasil:

```
{
  _id: ObjectId("6475511755638321bee73189"),
  address: {
    building: '146',
    coord: [ -73.9973041, 40.7188698 ],
    street: 'Mulberry Street',
    zipcode: '10013'
  },
  borough: 'Manhattan',
  cuisine: 'Italian',
  grades: [
    {
      date: ISODate("2014-05-02T00:00:00.000Z"),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate("2013-03-14T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2012-09-26T00:00:00.000Z"),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate("2012-02-15T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2011-09-15T00:00:00.000Z"),
      grade: 'A',
      score: 11
    }
  ],
  name: 'Angelo Of Mulberry St.',
  restaurant_id: '40365293'
}
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.

- b. Menampilkan data dengan name === "Angelo Of Mulberry St." menggunakan operasi query **\$eq**
10. menghapus restoran dari koleksi berdasarkan kriteria tertentu (tentukan sendiri kriteria)?

Jawab:

Query:

```
db.data.deleteOne({
  _id: ObjectId("6475511755638321bee7313a"),
  name: "Malang",
});
```

Hasil:

```
db.data.deleteOne({
...  _id: ObjectId("6475511755638321bee7313a"),
...  _id: ObjectId("6475511755638321bee7313a"),
... });
});
{ acknowledged: true, deletedCount: 1 }
restoran>
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. Menghapus field dengan kriteria id ObjectId("6475511755638321bee7313a") dan name "Malang" menggunakan operasi **\$eq**
- c. **deleteOne()** berfungsi untuk menghapus satu document, yang pertama kali cocok dengan hasil query
11. Tampilkan restoran dengan ulasan terbanyak?

Jawab:

Query:

```
db.data.aggregate([
{
  $project: {
    name: 1,
    totalReviews: { $size: "$grades" }
  }
},
{
  $sort: {
    totalReviews: -1
  }
},
{
  $limit: 1
}
]);
```

Hasil:

```
[
  {
    _id: ObjectId("6475511755638321bee7313f"),
    name: 'Ho Mei Restaurant',
    totalReviews: 8
  }
]
```

Note:

- Tahap pertama menggunakan \$project untuk menghitung jumlah ulasan pada setiap dokumen restoran, serta mempertahankan field "name" dalam output.
- Tahap kedua menggunakan \$sort untuk mengurutkan dokumen berdasarkan jumlah ulasan secara terurut menurun (descending), sehingga restoran dengan ulasan terbanyak akan berada di posisi pertama.
- Tahap ketiga menggunakan \$limit untuk membatasi hasil hanya satu dokumen.

12. Hitung jumlah restoran yang terletak di setiap negara bagian?

Jawab:

Query:

```
db.data.aggregate([
  {
    $group: { _id: "$borough", restaurant: { $sum : 1 } },
  },
]);
```

Hasil:

```
[
  { _id: 'Bronx', restaurant: 309 },
  { _id: 'Manhattan', restaurant: 1883 },
  { _id: 'Queens', restaurant: 738 },
  { _id: 'Staten Island', restaurant: 158 },
  { _id: 'Brooklyn', restaurant: 684 }
]
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- db.data.aggregate([...])**: Perintah ini digunakan untuk memulai operasi agregasi pada koleksi "data" di database yang aktif. Perintah ini akan menjalankan serangkaian operasi agregasi yang didefinisikan di dalam array [...].
- { \$group: { _id: "\$borough", restaurant: { \$sum : 1 } }}**: Ini adalah salah satu tahap dalam operasi agregasi yang menggunakan operator \$group. Operator \$group digunakan untuk mengelompokkan dokumen berdasarkan suatu kriteria. Dalam contoh ini, kita mengelompokkan dokumen berdasarkan nilai pada field "borough".
- _id: "\$borough"**: Ini adalah kunci yang digunakan untuk mengelompokkan dokumen. Dalam contoh ini, kita menggunakan nilai dari field "borough" sebagai kunci pengelompokan. Setiap grup akan memiliki nilai kunci ini.

- e. `restaurant: { $sum : 1 }`: Ini adalah ekspresi agregasi yang menghitung jumlah dokumen dalam setiap grup. Dalam contoh ini, kita menggunakan operator `$sum` untuk menghitung jumlah dokumen dalam setiap grup, dan memberikan nama field hasilnya sebagai "restaurant".

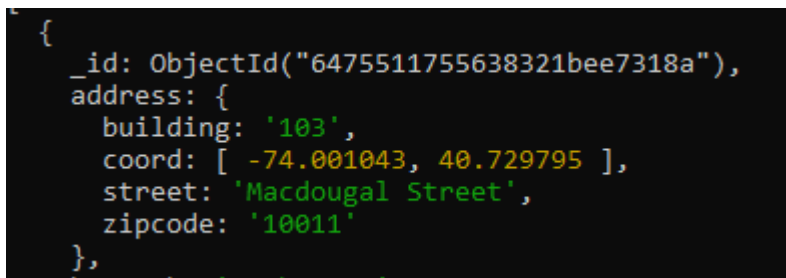
13. Pada restoran dengan ID "40365348", ganti nilai "zipcode" menjadi "10011".

Jawab:

Query:

```
db.data.updateOne(  
  {  
    restaurant_id: "40365348",  
  },  
  {  
    $set: {  
      "address.zipcode": "10011",  
    },  
  }  
);
```

Hasil:



```
{  
  _id: ObjectId("6475511755638321bee7318a"),  
  address: {  
    building: '103',  
    coord: [ -74.001043, 40.729795 ],  
    street: 'Macdougall Street',  
    zipcode: '10011'  
  },  
  ...  
}
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- Update field zipcode pada restaurant_id "40365348" menjadi "10011"
- \$set** berfungsi untuk mengubah nilai baru sesuai dengan hasil query sebelumnya

14. Berapa skor terakhir yang diberikan kepada restoran dengan ID "40365355"?

Jawab:

Query:

```
db.data.aggregate([  
  {  
    $match: {  
      restaurant_id: "40365355"  
    }  
  },  
  {  
    $project: {  
      lastScore: {  
        $arrayElemAt: ["$grades.score", -1]  
      }  
    }  
  }  
])
```

```

    }
  }
}
])

```

Hasil:

```

.. ])
{ _id: ObjectId("6475511755638321bee7318b"), lastScore: 12 } ]
restoran> db.data.find({

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. **{ \$match: { restaurant_id: "40365355" } }**: Ini adalah tahap pertama dalam operasi agregasi menggunakan operator **\$match**. Operator **\$match** digunakan untuk memfilter dokumen berdasarkan kriteria tertentu. Dalam contoh ini, kita memfilter dokumen yang memiliki nilai "restaurant_id" sama dengan "40365355". Hanya dokumen-dokumen yang memenuhi kriteria ini yang akan diproses dalam tahap-tahap agregasi berikutnya.
- c. **{ \$project: { lastScore: { \$arrayElemAt: ["\$grades.score", -1] } } }**: Ini adalah tahap kedua dalam operasi agregasi menggunakan operator **\$project**. Operator **\$project** digunakan untuk mengubah struktur dokumen dan menampilkan atau menghilangkan field-field tertentu. Dalam contoh ini, kita membuat field baru yang disebut "lastScore" dengan menggunakan operator **\$arrayElemAt**.
- d. **\$arrayElemAt: ["\$grades.score", -1]**: Operator **\$arrayElemAt** digunakan untuk mengakses elemen dalam array. Dalam contoh ini, kita mengakses elemen terakhir dari array "grades.score" yang terdapat di setiap dokumen. Nilai -1 digunakan untuk mengakses elemen terakhir dalam array
15. Pada restoran dengan ID "40365355", tambahkan nilai "score" sebesar 5 ke dalam array "grades".

Jawab:

Query:

```

db.data.updateOne(
{
restaurant_id: "40365355",
},
{
  $push: {
    grades: {
      date: ISODate(),
      grade: "A",
      score: 5,
    }
  },
}
);

```

Hasil:

```

grades: [
  {
    date: ISODate("2014-12-04T00:00:00.000Z"),
    grade: 'A',
    score: 11
  },
  {
    date: ISODate("2014-02-19T00:00:00.000Z"),
    grade: 'A',
    score: 10
  },
  {
    date: ISODate("2013-07-09T00:00:00.000Z"),
    grade: 'A',
    score: 9
  },
  {
    date: ISODate("2012-06-06T00:00:00.000Z"),
    grade: 'A',
    score: 10
  },
  {
    date: ISODate("2011-12-19T00:00:00.000Z"),
    grade: 'A',
    score: 12
  },
  {
    date: ISODate("2023-07-13T22:27:14.813Z"),
    grade: 'A',
    score: 5
  }
],
name: 'New Corner',
restaurant_id: '40365355',

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- updateOne()**, Berfungsi untuk update document sesuai dengan query
- update document dengan menambahkan field date, grade, score menggunakan array update operator **\$push()** Berfungsi untuk menambahkan elemen ke array
- Sesuai dengan restaurant_id

16. Tampilkan 10 restoran teratas berdasarkan rating dan jumlah ulasan?

Jawab:

Query:

```

db.data.aggregate([
  {
    $project: {
      name: 1,
      rating: { $avg: "$grades.score" },
      reviews: { $size: "$grades" }
    }
  },

```

```
{
  $sort: {
    rating: -1,
    reviews: -1
  }
},
{
  $limit: 10
}
]);
```

Hasil:

```
{
  name: 'Two Boots Grand Central',
  rating: 31,
  reviews: 5
},
{
  _id: ObjectId("6475511755638321bee732a5"),
  name: 'Victoria Pizza',
  rating: 30.8,
  reviews: 5
},
{
  _id: ObjectId("6475511755638321bee73e40"),
  name: 'Trinidad Golden Place',
  rating: 30.8,
  reviews: 5
},
{
  _id: ObjectId("6475511755638321bee73a12"),
  name: 'Billy'S Sport Bar Restaurant & Lounge',
  rating: 30.6,
  reviews: 5
},
{
  _id: ObjectId("6475511755638321bee73998"),
  name: 'La Candela Espanola',
  rating: 30,
  reviews: 3
}
}
```

Note:

- Tahap pertama menggunakan \$project untuk menghitung rata-rata rating dan jumlah ulasan pada setiap dokumen, serta mempertahankan field "name" dalam output.
- Tahap kedua menggunakan \$sort untuk mengurutkan dokumen berdasarkan rating secara terurut menurun (descending) dan jumlah ulasan secara terurut menurun (descending).
- Tahap ketiga menggunakan \$limit untuk membatasi hasil hanya 10 dokumen teratas.

17. Tampilkan restoran yang buka pada hari tertentu dengan jam operasional tertentu?

Jawab: -

Query: -

Hasil: -

Note: -

18. Pada restoran dengan ID "40365361", ganti nilai "cuisine" menjadi "Italian-American".

Jawab:

Query:

```
db.data.updateOne(
  {
    restaurant_id: "40365361",
  },
  {
    $set: {
      cuisine: "Italian-American",
    },
  }
);
```

Hasil:

```
{
  _id: ObjectId("6475511755638321bee7318c"),
  address: {
    building: '15',
    coord: [ -73.98126069999999, 40.7547107 ],
    street: 'West 43 Street',
    zipcode: '10036'
  },
  borough: 'Manhattan',
  cuisine: 'Italian-American',
  grades: [
    {
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- updateOne()**, Berfungsi untuk update document sesuai dengan query
- update sesuai dengan restaurant_id, lalu gunakan fungsi **\$set** untuk mengubah nilai field

19. Berapa jumlah elemen dalam array "grades" pada restoran dengan ID "40365387"?

Jawab:

Query:

```
db.data.aggregate([
  {
    $match: {
      restaurant_id: "40365387"
    }
  },
  {
    $project: {
      gradesCount: { $size: "$grades" }
    }
  }
])
```

```

    }
  }
});

```

Hasil:

```

.. });
);

.. [ { _id: ObjectId("6475511755638321bee7318d"), gradesCount: 5 } ]
restoran>

```

Note:

- Tahap pertama menggunakan \$match untuk memfilter dokumen yang memiliki "restaurant_id" yang sama dengan "40365387".
- Tahap kedua menggunakan \$project untuk menambahkan field "gradesCount" yang menggunakan operator \$size untuk menghitung jumlah elemen dalam array "grades"

20. Menghapus ulasan tertentu dari dokumen restoran?

Jawab:

Query:

```

db.data.deleteOne({
  "grades.grade": {
    $eq: "B"
  }
});

```

Hasil:

```

seq: "B"
.. }

.. });
);

acknowledged: true, deletedCount: 1 }
restoran>

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- deleteOne()** berfungsi untuk menghapus satu document, yang pertama kali cocok dengan hasil query
- hasilnya akan menghapus document yang mempunyai field grades, grade === "B"

21. Membuat indeks pada koleksi "restaurants" untuk meningkatkan kinerja pencarian?

Jawab:

Query:

```

db.data.createIndex({
  restaurant_id: 1
});

```

Hasil:

```

restoran> [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { restaurant_id: 1 }, name: 'restaurant_id_1' }
]
restoran>

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- membuat index di field `restaurant_id`

22. Tampilkan semua restoran yang berada di borough Manhattan.

Jawab:

Query:

```

db.data.find({
  borough: {
    $eq: "Manhattan"
  }
});

```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee73164"),
  address: {
    building: '1028',
    coord: [ -73.966032, 40.762832 ],
    street: '3 Avenue',
    zipcode: '10065'
  },
  borough: 'Manhattan',
  cuisine: 'Italian',
  grades: [
    {
      date: ISODate("2014-09-16T00:00:00.000Z"),
      grade: 'A',

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- Hasilnya menampilkan restoran yang berada di “Manhattan” Menggunakan operasi query **\$eq**

23. Berapa jumlah restoran yang memiliki kategori masakan "Italian"?

Jawab:

Query:

```

db.data.find({
  cuisine: {
    $eq: "Italian"
  }
}).count()

```

Hasil:

```
...
... 325
restoran>
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
 - Memfilter data dengan menampilkan data field cuisine === "Italian" menggunakan operasi **\$eq**
 - count()**, menampilkan jumlah hasil query
24. Tampilkan restoran dengan nama yang mengandung kata "Pizza".

Jawab:

Query:

```
db.data.find({
  name: {
    $regex: /Pizza/,
  },
});
```

Hasil:

```
    date: ISODate("2012-05-21T00:00:00.000Z"),
    grade: 'C',
    score: 68
  },
],
name: 'Victoria Pizza',
restaurant_id: '40374268'
}
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
 - Menampilkan semua field name yang didalamnya terdapat kata "Pizza" menggunakan operasi **\$regex**
25. Berapa jumlah restoran yang memiliki skor (score) lebih dari 10?

Jawab:

Query:

```
db.data.find({
  "grades.score": {
    $gt: 10,
  },
}).count()
```

Hasil:

```
3429
restoran>
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.

- b. Menggunakan operator **\$gt** ,berfungsi untuk memfilter data yang mempunyai nilai lebih dari
- c. **count()**, menampilkan jumlah hasil query

26. Tampilkan restoran yang memiliki skor (score) tertinggi.

Jawab:

Query:

`db.data.find().sort({ "grades.score": -1 }).limit(1);`

Hasil:

```

    date: ISODate("2014-03-28T00:00:00.000Z"),
    grade: 'C',
    score: 131
  },
  {
    date: ISODate("2013-09-25T00:00:00.000Z"),
    grade: 'A',
    score: 11
  },
  {
    date: ISODate("2013-04-08T00:00:00.000Z"),
    grade: 'B',
    score: 25
  },
  {
    date: ISODate("2012-10-15T00:00:00.000Z"),
    grade: 'A',
    score: 11
  },
  {
    date: ISODate("2011-10-19T00:00:00.000Z"),
    grade: 'A',
    score: 13
  }
],
name: "Murals On 54/Randolphs'S",
restaurant_id: '40372466'
}
estoran>

```

27. Berapa ium

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. `find()` digunakan untuk mencari semua dokumen di dalam koleksi "data".
- c. `sort({ "grades.score": -1 })` digunakan untuk mengurutkan dokumen berdasarkan skor (score) dalam array "grades" secara terurut menurun (descending), sehingga dokumen dengan skor tertinggi akan berada di posisi pertama.
- d. `limit(1)` digunakan untuk membatasi hasil yang ditampilkan hanya satu dokumen.

27. Berapa jumlah restoran yang berada di setiap borough?

Jawab:

Query:

```
db.data.aggregate([
  {
    $group: { _id: "$borough", restaurant: { $sum : 1 } },
  },
]);
```

Hasil:

```
[
  { _id: 'Bronx', restaurant: 309 },
  { _id: 'Manhattan', restaurant: 1883 },
  { _id: 'Queens', restaurant: 738 },
  { _id: 'Staten Island', restaurant: 158 },
  { _id: 'Brooklyn', restaurant: 684 }
]
```

Note:

- a. **db.data.aggregate([...])**: Perintah ini digunakan untuk memulai operasi agregasi pada koleksi "data" di database yang aktif. Perintah ini akan menjalankan serangkaian operasi agregasi yang didefinisikan di dalam array [...].
- b. **{ \$group: { _id: "\$borough", restaurant: { \$sum : 1 } } }**: Ini adalah salah satu tahap dalam operasi agregasi yang menggunakan operator \$group. Operator \$group digunakan untuk mengelompokkan dokumen berdasarkan suatu kriteria. Dalam contoh ini, kita mengelompokkan dokumen berdasarkan nilai pada field "borough".
- c. **_id: "\$borough"**: Ini adalah kunci yang digunakan untuk mengelompokkan dokumen. Dalam contoh ini, kita menggunakan nilai dari field "borough" sebagai kunci pengelompokan. Setiap grup akan memiliki nilai kunci ini.
- d. **restaurant: { \$sum : 1 }**: Ini adalah ekspresi agregasi yang menghitung jumlah dokumen dalam setiap grup. Dalam contoh ini, kita menggunakan operator \$sum untuk menghitung jumlah dokumen dalam setiap grup, dan memberikan nama field hasilnya sebagai "restaurant".

28. Tampilkan restoran dengan rating (score) di atas 10.

Jawab:

Query:

```
db.data.find({
  "grades.score": { $gt: 10 }
});
```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee73160"),
  address: {
    building: '277',
    coord: [ -73.8941893, 40.8634684 ],
    street: 'East Kingsbridge Road',
    zipcode: '10458'
  },
  borough: 'Bronx',
  cuisine: 'Chinese',
  grades: [
    {
      date: ISODate("2014-03-03T00:00:00.000Z"),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate("2013-09-26T00:00:00.000Z"),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate("2013-03-19T00:00:00.000Z"),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate("2012-08-29T00:00:00.000Z"),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate("2011-08-17T00:00:00.000Z"),
      grade: 'A',
      score: 13
    }
  ],
  name: 'Happy Garden',
  restaurant_id: '40364296'
}
]

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- "grades.score": { \$gt: 10 } adalah kriteria pencarian yang mencocokkan restoran yang memiliki rating (score) di atas 10. Operator \$gt digunakan untuk membandingkan nilai dan mencocokkan nilai yang lebih besar dari 10.
- find() digunakan untuk mencari dokumen di dalam koleksi "data" berdasarkan kriteria pencarian yang diberikan

29. Tampilkan restoran yang berada di zipcode "10013" dan memiliki grade (grade) "A"

Jawab:

Query:

db.data.find({

```

$and: [
  {
    "address.zipcode": {
      $eq: "10013",
    },
  },
  {
    "grades.grade": {
      $eq: "A",
    },
  },
],
});

```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee733f8"),
  address: {
    building: '59',
    coord: [ -74.0034907, 40.722135 ],
    street: 'Grand Street',
    zipcode: '10013'
  },
  borough: 'Manhattan',
  cuisine: 'American ',
  grades: [
    {
      date: ISODate("2014-05-20T00:00:00.000Z"),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate("2013-09-16T00:00:00.000Z"),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate("2013-03-23T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2012-06-25T00:00:00.000Z"),
      grade: 'A',
      score: 10
    }
  ],
}

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- \$and**: Operator \$and digunakan untuk menggabungkan beberapa kriteria pencarian. Dalam contoh ini, kita menggunakan \$and untuk menggabungkan dua kriteria pencarian.

- c. "address.zipcode": { \$eq: "10013" }: Ini adalah kriteria pencarian pertama yang menggunakan operator \$eq (equal) untuk mencocokkan nilai yang tepat. Kriteria ini mencari dokumen yang memiliki field "address.zipcode" dengan nilai "10013".
- d. "grades.grade": { \$eq: "A" }: Ini adalah kriteria pencarian kedua yang juga menggunakan operator \$eq untuk mencocokkan nilai yang tepat. Kriteria ini mencari dokumen yang memiliki field "grades.grade" dengan nilai "A".

30. Tampilkan restoran dengan rata-rata rating (score) di atas 8.

Jawab:

Query:

```
db.data.aggregate([
  {
    $group: { _id: "$name", rata_rata: { $avg: { $avg: "$grades.score" } } },
  }, {
    $match: {
      rata_rata: { $gt: 8 }
    }
  }
])
```

Hasil:

```
[
  { _id: 'Esperanto', rata_rata: 24 },
  { _id: 'Housing Works Food', rata_rata: 11 },
  { _id: 'Milanes Spanish Restaurant', rata_rata: 12.714285714285714 },
  {
    _id: 'Kum Gang San Korean Restaurant',
    rata_rata: 14.333333333333334
  },
  { _id: 'Gramercy Cafe', rata_rata: 10.166666666666666 },
  { _id: 'Society Of Illustrators', rata_rata: 11.4 },
  { _id: 'T-Bar Steak & Lounge', rata_rata: 9.166666666666666 },
  { _id: 'Cafe Loup', rata_rata: 12.6 },
  { _id: 'El Aguila Bakery', rata_rata: 13 },
  { _id: 'E.A.T. Cafe', rata_rata: 11.714285714285714 },
  { _id: 'Bartow Pizza', rata_rata: 13 },
  { _id: 'Scala Bakery', rata_rata: 8.833333333333334 },
  { _id: 'Sweet Life Cafe', rata_rata: 9 },
  { _id: 'Barney Greengrass', rata_rata: 11 },
  {
    _id: 'Cipriani Downtown Restaurant',
    rata_rata: 10.666666666666666
  },
  { _id: 'O'Flanagan'S', rata_rata: 11 },
  { _id: 'Sakagura', rata_rata: 10.666666666666666 },
  { _id: 'Cafe Bar', rata_rata: 15.2 },
  { _id: 'Strictly Vegetarian', rata_rata: 12 },
  { _id: 'Hyo Dong Gak', rata_rata: 9 }
]
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. { \$group: { _id: "\$name", rata_rata: { \$avg: { \$avg: "\$grades.score" } } } }:
- c. \$group: Operator \$group digunakan untuk mengelompokkan dokumen berdasarkan kriteria tertentu.

- d. `_id: "$name"`: Menentukan kunci pengelompokan. Kita mengelompokkan dokumen berdasarkan nilai pada field "name".
- e. `rata_rata: { $avg: { $avg: "$grades.score" } }`: Menghitung rata-rata nilai dari field "grades.score" dalam setiap grup. Operator `$avg` digunakan dua kali untuk menghitung rata-rata nilai rata-rata.
- f. `{ $match: { rata_rata: { $gt: 8 } } }`:
- g. `$match`: Operator `$match` digunakan untuk memfilter dokumen berdasarkan kriteria tertentu.
- h. `rata_rata: { $gt: 8 }`: Memfilter dokumen berdasarkan nilai "rata_rata" yang lebih besar dari 8.
- i. Jadi, perintah tersebut akan menghasilkan hasil agregasi yang akan mengelompokkan dokumen berdasarkan nilai pada field "name" dan menghitung rata-rata nilai dari field "grades.score" dalam setiap grup. Setelah itu, hanya dokumen-dokumen yang memiliki rata-rata nilai lebih besar dari 8 yang akan dikembalikan sebagai hasil akhir.

31. Tampilkan restoran yang memiliki kategori masakan "Mexican" di borough Manhattan.

Jawab:

Query:

```
db.data.find({
  $and: [
    {
      borough: {
        $eq: "Manhattan",
      },
    },
    {
      cuisine: {
        $eq: "Mexican",
      },
    },
  ],
});
```

Hasil:

```

_id: ObjectId("6475511755638321bee738ac"),
address: {
  building: '1470',
  coord: [ -73.9536119, 40.7705841 ],
  street: '1 Avenue',
  zipcode: '10075'
},
borough: 'Manhattan',
cuisine: 'Mexican',
grades: [
  {
    date: ISODate("2014-04-18T00:00:00.000Z"),
    grade: 'A',
    score: 13
  },
  {
    date: ISODate("2013-10-17T00:00:00.000Z"),
    grade: 'B',
    score: 14
  },
  {
    date: ISODate("2013-03-20T00:00:00.000Z"),
    grade: 'A',
    score: 12
  },
  {
    date: ISODate("2012-07-11T00:00:00.000Z"),
    grade: 'A',
    score: 12
  },
  {
    date: ISODate("2012-01-05T00:00:00.000Z"),
    grade: 'A',
    score: 2
  }
],
name: 'Canyon Road Grill',
restaurant_id: '40573131'

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
 - b. **\$and**: Operator **\$and** digunakan untuk menggabungkan beberapa kriteria pencarian. Dalam contoh ini, kita menggunakan **\$and** untuk menggabungkan dua kriteria pencarian.
 - c. **"borough": { \$eq: "Manhattan" }**: Ini adalah kriteria pencarian pertama yang menggunakan operator **\$eq** (equal) untuk mencocokkan nilai yang tepat. Kriteria ini mencari dokumen yang memiliki field **"borough"** dengan nilai **"Manhattan"**.
 - d. **"cuisine": { \$eq: "Mexican" }**: Ini adalah kriteria pencarian kedua yang juga menggunakan operator **\$eq** untuk mencocokkan nilai yang tepat. Kriteria ini mencari dokumen yang memiliki field **"cuisine"** dengan nilai **"Mexican"**.
32. Berapa jumlah restoran yang memiliki kategori masakan **"American"** di borough Brooklyn?

Jawab:

Query:

```
db.data.find({
  $and: [
    {
      borough: {
        $eq: "Brooklyn",
      },
    },
    {
      cuisine: {
        $eq: "American ",
      },
    },
  ],
}).count()
```

Hasil:

```
restoran> db.data.find({
...   $and: [
...     {
...       borough: {
...         $eq: "Brooklyn",
...       },
...     },
...     {
...       cuisine: {
...         $eq: "American ",
...       },
...     },
...   ],
... }).count()
189
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
 - b. **\$and**: Operator **\$and** digunakan untuk menggabungkan beberapa kriteria pencarian.
 - c. **{ borough: { \$eq: "Brooklyn" } }**: Kriteria pertama adalah "borough" harus sama dengan "Brooklyn". Operator **\$eq** digunakan untuk mencocokkan nilai yang tepat.
 - d. **{ cuisine: { \$eq: "American " } }**: Kriteria kedua adalah "cuisine" harus sama dengan "American ". Operator **\$eq** digunakan untuk mencocokkan nilai yang tepat.
 - e. Query ini akan mengembalikan dokumen-dokumen yang memenuhi kedua kriteria tersebut.
 - f. Terakhir, metode **.count()** digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria tersebut.
33. Pada restoran dengan ID "40365632", ganti nilai "street" menjadi "18th Avenue, Brooklyn".

Jawab:**Query:**


```

db.data.updateOne(
  {
    restaurant_id: "40365632",
  },
  {
    $set: {
      "address.street": "18th Avenue, Brooklyn",
    },
  }
);

```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee73194"),
  address: {
    building: '6322',
    coord: [ -73.9896898, 40.6199526 ],
    street: '18th Avenue, Brooklyn',
    zipcode: '11204'
  },
  borough: 'Brooklyn',
  cuisine: 'Pizza',
  grades: [
    {

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- `{ restaurant_id: "40365632" }`: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40365632". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
- `$set`: Operator `$set` digunakan untuk mengatur nilai field-field tertentu dalam dokumen yang akan diperbarui.
- `"address.street": "18th Avenue, Brooklyn"`: Ini adalah perubahan yang akan diterapkan pada dokumen. Dalam contoh ini, kita mengatur nilai field "address.street" menjadi "18th Avenue, Brooklyn".

34. Pada restoran dengan ID "40365632", hapus elemen terakhir dari array "grades"

Jawab:

Query:

```

db.data.updateOne(
  { restaurant_id: "40365632" },
  { $pop: { grades: 1 } }
)

```

Hasil:

```
.. {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
restoran>
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
 - b. { restaurant_id: "40365632" }: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40365632". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
 - c. \$pop: { grades: 1 }: Operator \$pop digunakan untuk menghapus elemen terakhir dari array dalam dokumen yang akan diperbarui. Dalam contoh ini, kita menggunakan \$pop pada field "grades" dengan nilai 1 untuk menghapus elemen terakhir dari array "grades" dalam dokumen.
35. Tampilkan restoran yang memiliki kategori masakan "Chinese" dan memiliki skor (score) di atas 8.

Jawab:

Query:

```
db.data.find({
  $and: [
    {
      "cuisine": {
        $eq: "Chinese",
      },
    },
    {
      "grades.score": {
        $gt: 8,
      },
    },
  ],
});
```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee7343f"),
  address: {
    building: '152',
    coord: [ -73.9771312, 40.67257840000001 ],
    street: '7 Avenue',
    zipcode: '11215'
  },
  borough: 'Brooklyn',
  cuisine: 'Chinese',
  grades: [
    {
      date: ISODate("2014-10-01T00:00:00.000Z"),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate("2014-03-11T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2013-02-06T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2012-01-26T00:00:00.000Z"),
      grade: 'A',
      score: 13
    }
  ],
  name: 'Szechuan Delight Restaurant',
  restaurant_id: '40391528'
}
]

```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. **\$and**: Operator **\$and** digunakan untuk menggabungkan beberapa kriteria pencarian. Dalam contoh ini, kita menggunakan **\$and** untuk menggabungkan dua kriteria pencarian.
- c. **"cuisine": { \$eq: "Chinese" }**: Ini adalah kriteria pencarian pertama yang menggunakan operator **\$eq** (equal) untuk mencocokkan nilai yang tepat. Kriteria ini mencari dokumen yang memiliki field "cuisine" dengan nilai "Chinese".
- d. **"grades.score": { \$gt: 8 }**: Ini adalah kriteria pencarian kedua yang menggunakan operator **\$gt** (greater than) untuk mencari nilai yang lebih besar dari 8 dalam array "grades.score". Kriteria ini mencari dokumen yang memiliki setidaknya satu elemen dalam array "grades.score" yang lebih besar dari 8.

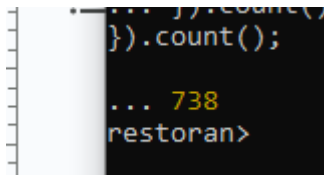
36. Berapa jumlah restoran yang buka pada hari Minggu di borough Queens?

Jawab:

Query:

```
db.data.find({
  $and: [
    { borough: "Queens" },
    { "hours.Sunday": { $ne: "Closed" } }
  ]
}).count();
```

Hasil:



```
... }).count();
... 738
restoran>
```

Note:

- \$and digunakan untuk menggabungkan dua atau lebih kriteria pencarian.
- { borough: "Queens" } adalah kriteria pencarian yang mencocokkan restoran dengan borough "Queens".
- { "hours.Sunday": { \$ne: "Closed" } } adalah kriteria pencarian yang mencocokkan restoran yang memiliki jam operasional pada hari Minggu, yaitu field "hours.Sunday" tidak sama dengan "Closed".
- find() digunakan untuk mencari dokumen di dalam koleksi "data" berdasarkan kriteria pencarian yang diberikan.
- .count() digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria pencarian.

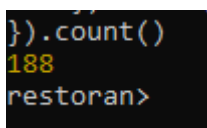
37. Berapa jumlah restoran dengan nama yang dimulai dengan huruf "A"?

Jawab:

Query:

```
db.data.find({
  name: {
    $regex: /^A/,
  },
}).count()
```

Hasil:



```
... }).count();
... 188
restoran>
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- name: { \$regex: /^A/ }: Ini adalah kriteria pencarian yang menggunakan operator \$regex untuk mencocokkan nilai field "name" dengan pola yang dimulai dengan huruf "A". Dalam contoh ini, pola pencarian adalah ^A, yang berarti mencari nilai "name" yang dimulai dengan huruf "A".
- count(): Metode .count() digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria pencarian.

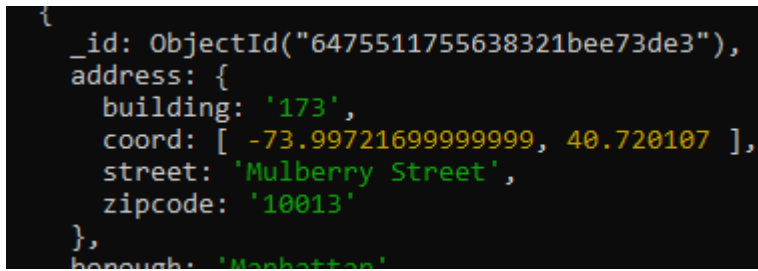
38. Tampilkan restoran yang memiliki alamat di "Mulberry Street".

Jawab:

Query:

```
db.data.find({
  "address.street": {
    $eq: "Mulberry Street",
  },
})
```

Hasil:



```
{
  _id: ObjectId("6475511755638321bee73de3"),
  address: {
    building: '173',
    coord: [ -73.99721699999999, 40.720107 ],
    street: 'Mulberry Street',
    zipcode: '10013'
  },
  borough: 'Manhattan'
}
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- "address.street": { \$eq: "Mulberry Street" }: Ini adalah kriteria pencarian yang menggunakan operator \$eq (equal) untuk mencocokkan nilai field "address.street" dengan "Mulberry Street". Kriteria ini mencari dokumen yang memiliki nilai field "address.street" yang sama dengan "Mulberry Street".

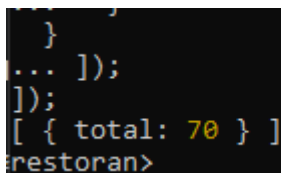
39. Berapa jumlah restoran dengan kategori masakan unik yang ada?

Jawab:

Query:

```
db.data.aggregate([
  {
    $group: {
      _id: "$cuisine"
    }
  },
  {
    $count: "total"
  }
]);
```

Hasil:



```
{
  _id: '...',
  total: 70
}
... ]);
]);
[ { total: 70 } ]
restaurant>
```

Note:

- Tahap pertama menggunakan \$group untuk mengelompokkan dokumen berdasarkan field "cuisine". Setiap kelompok akan memiliki field "_id" yang mewakili kategori masakan unik.

- b. Tahap kedua menggunakan \$count untuk menghitung jumlah kelompok yang terbentuk dan memberikan hasil akhir dalam field "total".

40. Tampilkan restoran yang terletak di borough Brooklyn dan memiliki kategori masakan tertentu

Jawab:

Query:

```
db.data.find({
  $and: [
    {
      borough: {
        $eq: "Brooklyn",
      },
    },
    {
      cuisine: {
        $eq: "Bakery",
      },
    },
  ],
});
```

Hasil:

```
building: 2214 ,
coord: [ -73.9936139, 40.601451 ],
street: '86 Street',
zipcode: '11214'
},
borough: 'Brooklyn',
cuisine: 'Bakery',
grades: [
  {
    date: ISODate("2013-12-12T00:00:00.000Z"),
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- \$and: Operator \$and digunakan untuk menggabungkan dua atau lebih kriteria pencarian.
- Kriteria pertama adalah "borough" harus sama dengan "Brooklyn". Kita menggunakan operator \$eq (equal) untuk mencocokkan nilai yang tepat.
- Kriteria kedua adalah "cuisine" harus sama dengan "Bakery". Kembali menggunakan operator \$eq untuk mencocokkan nilai yang tepat.

41. Tampilkan restoran dengan zipcode "11211" dan memiliki grade (grade) "A"

Jawab:

Query:

```
db.data.find({
  $and: [
    {
```

```

    "address.zipcode": {
      $eq: "11211",
    },
  },
  {
    "grades.grade": {
      $eq: "A",
    },
  },
],
});

```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee73c6d"),
  address: {
    building: '556',
    coord: [ -73.95676279999999, 40.7170508 ],
    street: 'Driggs Avenue',
    zipcode: '11211'
  },
  borough: 'Brooklyn',
  cuisine: 'Italian',
  grades: [
    {
      date: ISODate("2014-09-10T00:00:00.000Z"),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate("2013-08-19T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2013-05-14T00:00:00.000Z"),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate("2012-05-01T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2011-12-28T00:00:00.000Z"),
      grade: 'A',
      score: 13
    }
  ]
}

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- \$and**: Operator **\$and** digunakan untuk menggabungkan dua atau lebih kriteria pencarian.

- c. Kriteria pertama adalah "address.zipcode" harus sama dengan "11211". Kita menggunakan operator \$eq (equal) untuk mencocokkan nilai yang tepat.
- d. Kriteria kedua adalah "grades.grade" harus sama dengan "A". Kembali menggunakan operator \$eq untuk mencocokkan nilai yang tepat.

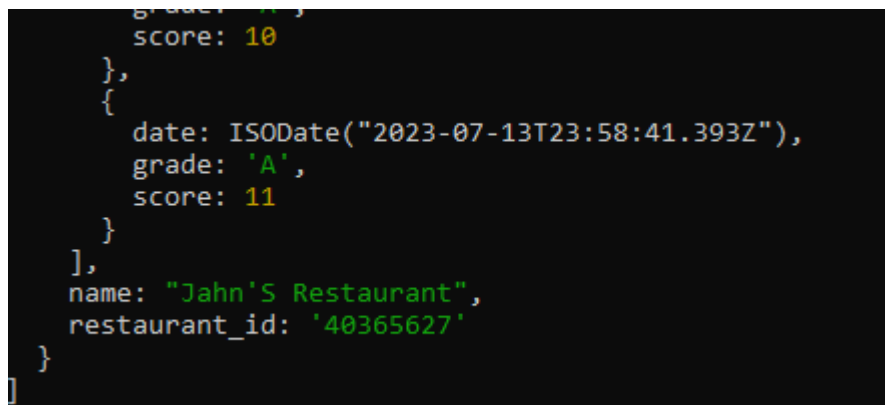
42. Pada restoran dengan ID "40365627", tambahkan nilai "grade" dan "score" baru ke dalam array "grades": {"grade": "A", "score": 11}.

Jawab:

Query:

```
db.data.updateOne(
  {
    restaurant_id: "40365627",
  },
  {
    $push: {
      grades: {
        date: ISODate(),
        grade: "A",
        score: 11,
      }
    }
  },
  {
  });
```

Hasil:



```
[
  {
    grades: [
      {
        date: ISODate("2023-07-13T23:58:41.393Z"),
        grade: 'A',
        score: 11
      }
    ],
    name: "Jahn'S Restaurant",
    restaurant_id: '40365627'
  }
]
```

Note:

- a. **db.data** mengacu pada koleksi pada database yang digunakan.
- b. { restaurant_id: "40365627" }: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40365627". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
- c. \$push: Operator \$push digunakan untuk menambahkan elemen baru ke dalam array "grades" dalam dokumen yang akan diperbarui.
- d. "grades": { date: ISODate(), grade: "A", score: 11 }: Ini adalah perubahan yang akan diterapkan pada dokumen. Dalam contoh ini, kita menggunakan \$push untuk menambahkan elemen baru ke dalam array "grades". Elemen baru ini memiliki field

"date" dengan nilai tanggal saat ini (ISODate()), field "grade" dengan nilai "A", dan field "score" dengan nilai 11.

43. Pada restoran dengan ID "40365627", ganti nilai "cuisine" menjadi "Chinese-American".

Jawab:

Query:

```
db.data.updateOne(
  {
    restaurant_id: "40365627",
  },
  {
    $set: {
      cuisine: "Chinese-American",
    },
  }
);
```

Hasil:

```
{
  _id: ObjectId("6475511755638321bee73193"),
  address: {
    building: '8104',
    coord: [ -73.8850023, 40.7494272 ],
    street: '37 Avenue',
    zipcode: '11372'
  },
  borough: 'Queens',
  cuisine: 'Chinese-American',
  grades: [
    {
      date: ISODate("2014-07-07T00:00:00.000Z"),
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- { restaurant_id: "40365627" }: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40365627". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
- \$set: Operator \$set digunakan untuk mengatur nilai field tertentu dalam dokumen yang akan diperbarui.
- "cuisine": "Chinese-American": Ini adalah perubahan yang akan diterapkan pada dokumen. Dalam contoh ini, kita menggunakan \$set untuk mengubah nilai field "cuisine" menjadi "Chinese-American"

44. Tampilkan restoran yang memiliki grade (grade) "A" pada tanggal 1 Januari 2014.

Jawab:

Query:

Hasil:

Note:

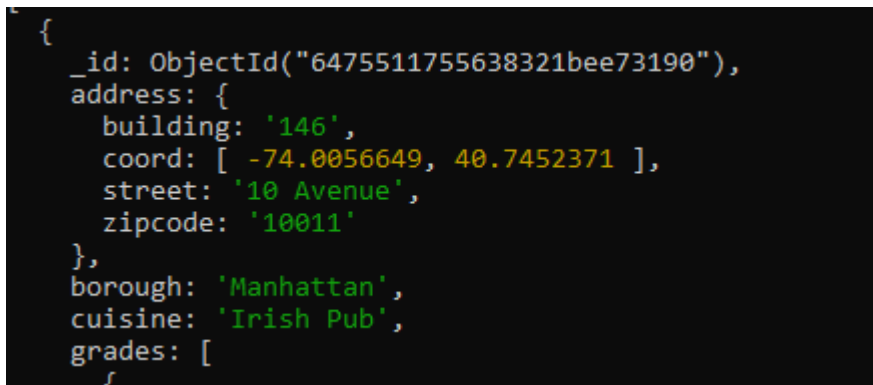
45. Pada restoran dengan ID "40365526", ganti nilai "cuisine" menjadi "Irish Pub".

Jawab:

Query:

```
db.data.updateOne(  
  {  
    restaurant_id: "40365526",  
  },  
  {  
    $set: {  
      cuisine: "Irish Pub",  
    },  
  }  
);
```

Hasil:



```
{  
  _id: ObjectId("6475511755638321bee73190"),  
  address: {  
    building: '146',  
    coord: [ -74.0056649, 40.7452371 ],  
    street: '10 Avenue',  
    zipcode: '10011'  
  },  
  borough: 'Manhattan',  
  cuisine: 'Irish Pub',  
  grades: [  
    {
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- `{ restaurant_id: "40365526" }`: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40365526". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
- `$set`: Operator `$set` digunakan untuk mengatur nilai field tertentu dalam dokumen yang akan diperbarui.
- `"cuisine": "Irish Pub"`: Ini adalah perubahan yang akan diterapkan pada dokumen. Dalam contoh ini, kita menggunakan `$set` untuk mengubah nilai field "cuisine" menjadi "Irish Pub".

46. Hapus elemen pertama dari array "grades" pada restoran tertentu.

Jawab:

Query:

```
db.data.updateOne(  
  {  
    restaurant_id : "40356018"  
  },  
  {  
    $pop: {
```

```
    grades: -1,  
  },  
}  
)
```

Hasil:

```
{  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- { restaurant_id: "40356018" }: Ini adalah kriteria pencarian yang menggunakan field "restaurant_id" dengan nilai "40356018". Perintah ini akan mencari dokumen dengan "restaurant_id" yang sesuai dengan kriteria ini.
- \$pop: Operator \$pop digunakan untuk menghapus elemen terakhir dari array "grades" dalam dokumen yang akan diperbarui.
- "grades": -1: Ini adalah perubahan yang akan diterapkan pada dokumen. Dalam contoh ini, kita menggunakan \$pop dengan nilai -1 untuk menghapus elemen terakhir dari array "grades" dalam dokumen.

47. Tampilkan restoran dengan nama yang diawali huruf "A".

Jawab:

Query:

```
db.data.find({  
  name: {  
    $regex: /^A/  
  }  
});
```

Hasil:

```

    score: 10
  },
  {
    date: ISODate("2014-05-16T00:00:00.000Z"),
    grade: 'A',
    score: 11
  },
  {
    date: ISODate("2013-09-05T00:00:00.000Z"),
    grade: 'A',
    score: 12
  },
  {
    date: ISODate("2013-03-05T00:00:00.000Z"),
    grade: 'A',
    score: 11
  },
  {
    date: ISODate("2012-04-12T00:00:00.000Z"),
    grade: 'A',
    score: 12
  }
],
name: 'Arirang Hibachi Steak House',
restaurant_id: '40386058'

```

Note:

- a. { name: { \$regex: /^A/ } }: Ini adalah kriteria pencarian yang menggunakan operator \$regex untuk mencocokkan pola dalam nilai field "name". Dalam contoh ini, /^A/ berarti mencocokkan nama yang diawali dengan huruf "A".
- b. find() digunakan untuk mencari dokumen di dalam koleksi "data" berdasarkan kriteria pencarian yang diberikan.

48. Tampilkan restoran yang memiliki rata-rata skor (score) tertinggi.

Jawab:

Query:

```

db.data.aggregate([
  {
    $unwind: "$grades"
  },
  {
    $group: {
      _id: "$_id",
      name: { $first: "$name" },
      averageScore: { $avg: "$grades.score" }
    }
  },
  {
    $sort: {
      averageScore: -1
    }
  }
])

```

```

    }
  },
  {
    $limit: 1
  }
]);

```

Hasil:

```

.. [
  {
    _id: ObjectId("6475511755638321bee73488"),
    name: 'Bella Napoli',
    averageScore: 38.6
  }
]

```

Note:

- Tahap pertama menggunakan \$unwind untuk memisahkan setiap elemen dalam array "grades" menjadi dokumen tersendiri.
- Tahap kedua menggunakan \$group untuk mengelompokkan dokumen berdasarkan field "_id" (dalam hal ini, field "_id" adalah _id dari dokumen asli) dan menghitung rata-rata skor (score) dari array "grades" menggunakan operator \$avg.
- Tahap ketiga menggunakan \$sort untuk mengurutkan dokumen berdasarkan rata-rata skor (score) secara terurut menurun (descending), sehingga dokumen dengan rata-rata skor tertinggi akan berada di posisi pertama.
- Tahap keempat menggunakan \$limit dengan nilai 1 untuk membatasi hasil yang ditampilkan hanya satu dokumen dengan rata-rata skor tertinggi.

49. Berapa jumlah restoran dengan "cuisine" "Italian"?

Jawab:

Query:

```

db.data.find({
  cuisine: {
    $eq: "Italian",
  },

```

```

}).count()

```

Hasil:

```

..
.. }).count()
325
restoran>

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.

- b. { cuisine: { \$eq: "Italian" } } : Ini adalah kriteria pencarian yang menggunakan operator \$eq (equal) untuk mencocokkan nilai field "cuisine" dengan "Italian". Kriteria ini mencari dokumen yang memiliki nilai field "cuisine" yang sama dengan "Italian".
- c. .count(): Metode .count() digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria pencarian

50. Tampilkan restoran yang memiliki ulasan (grades) dengan skor (score) terendah.

Jawab:

Query:

```
db.data.aggregate([
  {
    $unwind: "$grades"
  },
  {
    $sort: {
      "grades.score": 1
    }
  },
  {
    $group: {
      _id: "$_id",
      restaurant: { $first: "$$ROOT" }
    }
  },
  {
    $limit: 1
  }
]);
```

Hasil:

```

{
  _id: ObjectId("6475511755638321bee733ac"),
  restaurant: {
    _id: ObjectId("6475511755638321bee733ac"),
    address: {
      building: '138',
      coord: [ -73.99741519999999, 40.7186078 ],
      street: 'Mulberry Street',
      zipcode: '10013'
    },
    borough: 'Manhattan',
    cuisine: 'Italian',
    grades: {
      date: ISODate("2013-04-11T00:00:00.000Z"),
      grade: 'A',
      score: 11
    },
    name: "Pellegrino'S",
    restaurant_id: '40387086'
  }
}

```

Note:

- Tahap pertama menggunakan \$unwind untuk memisahkan setiap elemen dalam array "grades" menjadi dokumen tersendiri.
- Tahap kedua menggunakan \$sort untuk mengurutkan dokumen berdasarkan skor (score) dalam array "grades" secara terurut naik (ascending), sehingga dokumen dengan skor terendah akan berada di posisi pertama.
- Tahap ketiga menggunakan \$group untuk mengelompokkan dokumen berdasarkan field "_id" (dalam hal ini, field "_id" adalah _id dari dokumen asli) dan memilih dokumen pertama dalam setiap kelompok menggunakan operator \$first.
- Tahap keempat menggunakan \$limit dengan nilai 1 untuk membatasi hasil yang ditampilkan hanya satu dokumen dengan skor terendah.

51. Berapa jumlah restoran dengan kategori masakan "Pizza" di setiap borough?

Jawab:

Query:

```

db.data.aggregate([
  {
    $match: {
      cuisine: "Pizza"
    }
  },
  {
    $group: {
      _id: "$borough",
      count: { $sum: 1 }
    }
  }
])

```

```
});
```

Hasil:

```
.. [
  { _id: 'Queens', count: 80 },
  { _id: 'Bronx', count: 35 },
  { _id: 'Staten Island', count: 10 },
  { _id: 'Brooklyn', count: 72 },
  { _id: 'Manhattan', count: 73 }
]
restoran>
```

Note:

- Tahap pertama menggunakan `$match` untuk memfilter dokumen-dokumen yang memiliki kategori masakan "Pizza".
- Tahap kedua menggunakan `$group` untuk mengelompokkan dokumen-dokumen berdasarkan field "borough". Setiap kelompok akan memiliki field "_id" yang mewakili borough dan field "count" yang menunjukkan jumlah restoran dalam kelompok tersebut.

52. Tampilkan restoran dengan alamat yang mengandung kata "Avenue".

Jawab:

Query:

```
db.data.find({
  "address.street": {
    $regex: /Avenue/,
  },
});
```

Hasil:

```
{
  _id: ObjectId("6475511755638321bee73156"),
  address: {
    building: '7905',
    coord: [ -73.8740217, 40.7135015 ],
    street: 'Metropolitan Avenue',
    zipcode: '11379'
  },
  borough: 'Queens',
  cuisine: 'Bagels/Pretzels',
  grades: [
    {
      date: ISODate("2014-09-17T00:00:00.000Z"),
      grade: 'A',
    }
  ]
}
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- "address.street": { \$regex: /Avenue/ }: Ini adalah kriteria pencarian yang menggunakan operator `$regex` untuk mencocokkan pola dalam nilai field "address.street". Dalam contoh ini, kita mencari dokumen yang memiliki nilai field "address.street" yang mengandung kata "Avenue".

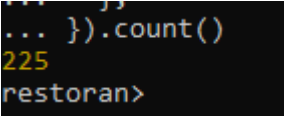
53. Berapa jumlah restoran yang memiliki nama yang mengandung kata "Bar"?

Jawab:

Query:

```
db.data.find({
  name: {
    $regex: /Bar/,
  },
}).count()
```

Hasil:



```
... }
... }).count()
225
restoran>
```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- "name": { \$regex: /Bar/ }: Ini adalah kriteria pencarian yang menggunakan operator \$regex untuk mencocokkan pola dalam nilai field "name". Dalam contoh ini, kita mencari dokumen yang memiliki nilai field "name" yang mengandung kata "Bar".
- .count(): Metode .count() digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria pencarian.

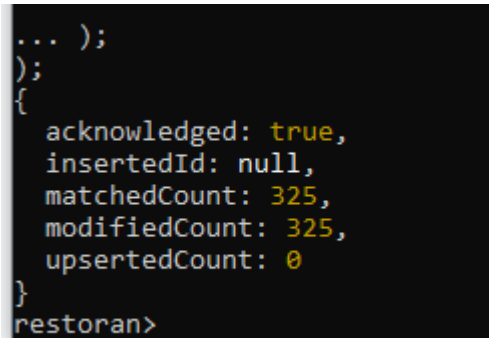
54. Pada semua restoran dengan "cuisine" "Italian", tambahkan nilai "score" sebesar 2 ke dalam setiap elemen array "grades".

Jawab

Query:

```
db.data.updateMany(
{
  cuisine: "Italian"
},
{
  $inc: {
    "grades.$.score": 2
  }
}
);
```

Hasil:



```
... );
);
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 325,
  modifiedCount: 325,
  upsertedCount: 0
}
restoran>
```

Note:

- a. updateMany() digunakan untuk memperbarui beberapa dokumen sekaligus dalam koleksi "data".
- b. { cuisine: "Italian" } adalah kriteria pencarian yang mencocokkan restoran dengan "cuisine" "Italian".
- c. \$inc digunakan untuk menambahkan nilai ke field yang ada dalam dokumen.
- d. "grades.\$[].score" digunakan untuk mengakses setiap elemen "score" dalam array "grades" pada dokumen dan menambahkan nilai 2.

e.

55. Tampilkan restoran yang memiliki ulasan (grades) dengan grade "A" di setiap borough.

Jawab:**Query:**

```
db.data.aggregate([
  {
    $match: {
      "grades.grade": "A"
    }
  },
  {
    $group: {
      _id: "$borough",
      restaurants: { $push: "$$ROOT" }
    }
  }
]);
```

Hasil:

```

    street: 'Parsons Boulevard',
    zipcode: '11433'
  },
  borough: 'Queens',
  cuisine: 'Caribbean',
  grades: [
    {
      date: ISODate("2014-04-10T00:00:00.000Z"),
      grade: 'A',
      score: 7
    },
    {
      date: ISODate("2013-11-01T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2013-05-23T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2012-12-06T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2011-11-09T00:00:00.000Z"),
      grade: 'A',
      score: 12
    }
  ]
}

```

Note:

- Tahap pertama menggunakan \$match untuk memfilter dokumen-dokumen yang memiliki grade "A" dalam array "grades".
- Tahap kedua menggunakan \$group untuk mengelompokkan dokumen-dokumen berdasarkan field "borough". Setiap kelompok akan memiliki field "_id" yang mewakili borough dan array "restaurants" yang berisi dokumen-dokumen restoran dalam kelompok tersebut.

56. Berapa jumlah restoran yang berada di zipcode 10013?

Jawab

Query:

```

db.data.find({
  "address.zipcode": {
    $eq: "10013",
  },
}).count()

```

Hasil:

```

... },
... }).count()
88
restoran>

```

Note:

- db.data** mengacu pada koleksi pada database yang digunakan.
- "address.zipcode": { \$eq: "10013" }: Ini adalah kriteria pencarian yang menggunakan operator \$eq (equal) untuk mencocokkan nilai field "address.zipcode" dengan "10013". Kriteria ini mencari dokumen yang memiliki nilai field "address.zipcode" yang sama dengan "10013".
- .count(): Metode .count() digunakan untuk menghitung jumlah dokumen yang memenuhi kriteria pencarian

57. Tampilkan restoran yang memiliki nama yang panjangnya lebih dari 20 karakter.

Jawab

Query:

```
db.data.find({
  $expr: { $gt: [{ $strLenCP: "$name" }, 20] }
});
```

Hasil:

```

    date: ISODate("2014-12-18T00:00:00.000Z"),
    grade: 'A',
    score: 7
  },
  {
    date: ISODate("2014-05-01T00:00:00.000Z"),
    grade: 'B',
    score: 17
  },
  {
    date: ISODate("2013-03-14T00:00:00.000Z"),
    grade: 'A',
    score: 12
  },
  {
    date: ISODate("2012-09-20T00:00:00.000Z"),
    grade: 'A',
    score: 9
  },
  {
    date: ISODate("2012-02-08T00:00:00.000Z"),
    grade: 'B',
    score: 19
  }
],
name: 'The New Starling Athletic Club Of The Bronx',
restaurant_id: '40364956'
}
```

Note:

- \$expr digunakan untuk mengevaluasi ekspresi MongoDB.
- \$strLenCP digunakan untuk menghitung panjang nama dalam karakter pada field "name".
- { \$gt: [{ \$strLenCP: "\$name" }, 20] } digunakan untuk membandingkan panjang nama dengan nilai 20, dan hanya mengembalikan dokumen yang panjang namanya lebih dari 20 karakter.

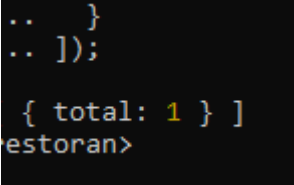
58. Berapa jumlah restoran dengan koordinat geografis unik yang ada?

Jawab

Query:

```
db.data.aggregate([
  {
    $group: {
      _id: { latitude: "$address.coord.0", longitude: "$address.coord.1" },
      count: { $sum: 1 }
    }
  },
  {
    $match: {
      count: { $gt: 1 }
    }
  },
  {
    $count: "total"
  }
]);
```

Hasil:



```
.. }
.. ]);

{ total: 1 } ]
estoran>
```

Note:

- Tahap pertama menggunakan \$group untuk mengelompokkan dokumen berdasarkan koordinat geografis pada field "address.coord". Kami menggunakan objek _id untuk mewakili pasangan koordinat latitude dan longitude.
- Tahap kedua menggunakan \$match untuk memfilter hanya kelompok yang memiliki lebih dari satu dokumen (artinya, koordinat geografis yang sama).
- Tahap ketiga menggunakan \$count untuk menghitung jumlah kelompok yang memenuhi kriteria sebelumnya dan memberikan hasil akhir dalam field "total".

59. Tampilkan restoran yang memiliki kategori masakan "Italian" dan skor (score) terendah.

Jawab

Query:

```
db.data.aggregate([
  {
    $match: {
      cuisine: "Italian"
    }
  },
  {
```

```
$sort: {
  "grades.score": 1
},
{
  $limit: 1
}
]);
```

Hasil:

```
borough: 'Staten Island',
cuisine: 'Italian',
grades: [
  {
    date: ISODate("2014-07-23T00:00:00.000Z"),
    grade: 'B',
    score: -1
  },
  {
    date: ISODate("2013-04-02T00:00:00.000Z"),
    grade: 'A',
    score: 13
  },
  {
    date: ISODate("2012-10-18T00:00:00.000Z"),
    grade: 'A',
    score: 10
  },
  {
    date: ISODate("2011-09-27T00:00:00.000Z"),
    grade: 'A',
    score: 10
  }
],
name: 'Cafe Bella Vita',
restaurant_id: '40392313'
}
```

Note:

- Tahap pertama menggunakan \$match untuk memfilter dokumen berdasarkan kategori masakan "Italian".
- Tahap kedua menggunakan \$sort untuk mengurutkan dokumen berdasarkan skor (score) dalam array "grades" secara terurut naik (ascending).
- Tahap ketiga menggunakan \$limit dengan nilai 1 untuk membatasi hasil yang ditampilkan hanya satu dokumen dengan skor terendah