2. Implementación de consultas simples

- Buscar todos los restaurantes de un tipo de comida específico (ej. "Chinese").

db.restaurants.find({type_of_food: "Sushi"})

Filtramos los documentos por el tipo de comida, en este caso Sushi, y al ejecutar esta consulta nos salen 4 restaurantes diferentes

Ejemplo de resultado:


```
{
  _id: ObjectId('55f14312c7447c3da7051c94'),
  URL: 'http://www.just-eat.co.uk/restaurants-aisushi-n12/menu',
  address: '830 High Road',
  'address line 2': 'London',
  name: 'Ai Sushi',
  outcode: 'N12',
  postcode: '9RA',
  rating: 4.5,
  type_of_food: 'Sushi'
}
{
  _id: ObjectId('55f14312c7447c3da7051c9f'),
  URL: 'http://www.just-eat.co.uk/restaurants-aisurusushi-nw1/menu',
  address: '112 Kentish Town Road',
  'address line 2': 'Regents Park',
  name: 'Aisuru Sushi',
  outcode: 'NW1',
  postcode: '9PX',
  rating: 5,
  type_of_food: 'Sushi'
}
```


Listar las inspecciones con violaciones, ordenadas por fecha.

db.inspections.find({result: "Violation Issued", }).sort({date: 1})

En primer lugar, observamos los diferentes documentos de la base de datos para ver como queda registrado el resultado de la inspección. Entre ellos vemos diferentes resultados: 'Violation Issued'**,** 'Warning Issued', 'Fail', 'Pass' o 'No Violation Issued'**.** De esta manera ahora ya sabemos como debe ser el string para filtrar la búsqueda.

En segundo lugar, utilizamos el find para filtrar las diferentes inspecciones y que cumplan la restricción marcada, es decir que el resultado de la inspección sea el valor consultado anteriormente ("Violation Issued").

Finalmente, añadimos el sort para así ordenar los documentos obtenidos por el campo de la fecha. Como queremos que estos vayan de más antiguas a más nueva, utilizamos el 1. Entonces, obtenemos como resultado las inspecciones ordenadas, por ejemplo:

```
_id: ObjectId('56d61034a378eccde8a8dfec'),
id: '71327-2015-ENFO',
certificate_number: 3048742,
business_name: 'ASHOKA',
date: 2022-03-06T23:00:00.000Z,
result: 'Violation Issued',
sector: 'Grocery-Retail - 808',
address: {
  city: 'EDINBURGH',
  zip: '1DJ',
  street: 'HANOVER STREET',
  number: '97'
},
restaurant_id: '55f14312c7447c3da7051fec'


_id: ObjectId('56d61035a378eccde8a93196'),
id: '42368-2015-ENFO',
certificate_number: 9304312,
business_name: 'ATHENA',
date: 2022-03-08T23:00:00.000Z,
result: 'Violation Issued',
sector: 'Salons And Barbershop - 841',
address: {
  city: 'GLASGOW',
  zip: '7AW',
  street: 'ELDERSLIE STREET',
  number: '141'
},
```


Encontrar restaurantes con una calificación superior a 4

db.restaurants.find({ $expr: { $gt: ["rating", 4] } })

En esta consulta de aquí utilizamos una expresión para comparar el valor del campo rating y el 4. Como tiene que ser mayor de 4 utilizamos el operador greater than (gt). Al ejecutar esto nos salen los siguientes resultados:

```
{
  _id: ObjectId('55f14312c7447c3da7051b38'),
  URL: 'http://www.just-eat.co.uk/restaurants-168chinese-ls18/menu',
  address: '17 Alexandra Road',
  'address line 2': 'West Yorkshire',
  name: '168 Chinese & Cantonese Takeaway',
  outcode: 'LS18',
  postcode: '4HE',
  rating: 5.5,
  type_of_food: 'Chinese'
}
{
  _id: ObjectId('55f14312c7447c3da7051b39'),
  URL: 'http://www.just-eat.co.uk/restaurants-1awok-pa7/menu',
  address: 'Unit 2 30 Greenock Road',
  'address line 2': 'Bishopton',
  name: '1A Wok',
  outcode: 'PA7',
  postcode: '5JN',
  rating: 5,
  type_of_food: 'Chinese'
}
```


Uso de agregaciones

- Agrupar restaurantes por tipo de comida y calcular la calificación promedio.

db.restaurants.aggregate([{ $match: { "rating": { $ne: "Not yet rated" } } },
{ $group: { _id: "$type_of_food", avgRating: { $avg: { $toDouble: "$rating" } }} }])

Hemos dividido esta agregación en dos partes. En primer lugar, eliminamos aquellos restaurantes que aún no han sido calificados. Para hacer esta operación aplicamos el **match**. En segundo lugar, **agrupamos** los diferentes resultados por el tipo de comida y calculamos un nuevo campo utilizando el operador **avg** para calcular la media de calificación. Además, para asegurar un resultado correcto, convertimos el valor del campo rating a double utilizando **toDouble**. Entonces, nos salen los siguientes resultados


```
{
  _id: 'Mediterranean',
  avgRating: 5.166666666666667
}
{
  _id: 'American',
  avgRating: 4.617021276595745
}
{
  _id: 'Chicken',
  avgRating: 4.41
}
{
  _id: 'Peri Peri',
  avgRating: 4.868421052631579
}
{
  _id: 'South Curry',
  avgRating: 4.833333333333333
}
{
  _id: 'Thai',
  avgRating: 4.65
}
```

- **Contar el número de inspecciones por resultado y mostrar los porcentajes.**

```
db.inspections.aggregate([
 { $setWindowFields: { output: { totalInspeccions: { $count: {} } } } },
{ $group: { _id: "$result", totalResultat: { $sum: 1 }, totalInspeccions: { $first: "$totalInspeccions" } } },
{ $addFields: { porcentaje: { $multiply: [{ $divide: ["$totalResultat", "$totalInspeccions"] }, 100] } } } ])
```

Podemos diferenciar tres etapas diferentes en esta agregación. En primer lugar, calculamos el total de Inspecciones guardadas en la base de datos utilizando el **count**.
En segundo lugar, **agrupamos** los resultados por el resultado de la inspección y contamos cuántas inspecciones tienen el mismo tipo de resultado. También, nos aseguramos de no perder el cálculo del total de inspecciones que hay.
Finalmente, con todos los datos ya calculados, hacemos las operaciones necesarias para calcular el porcentaje de inspecciones con cada resultado. A través de los resultados podemos ver que el resultado más común es "Violation Issued", seguido de "Fail" y "Warning Issued", luego "No Violation Issued" y, para acabar, "Pass"

```
{
  _id: 'Pass',
  totalResultat: 1259,
  totalInspeccions: 6370,
  porcentaje: 19.76452119309262
}
{
  _id: 'Fail',
  totalResultat: 1280,
  totalInspeccions: 6370,
  porcentaje: 20.09419152276295
}
{
  _id: 'Warning Issued',
  totalResultat: 1280,
  totalInspeccions: 6370,
  porcentaje: 20.09419152276295
}
```

- Unir restaurantes con sus inspecciones utilizando $lookup

Para hacer esta consulta utilizamos el operador **lookup**. Este consta de diferentes parámetros. En primer lugar, encontramos el "from", en este campo se especifica con qué colección se pretende fusionar.
Después, tenemos el "localField" donde se especifica que campo de la colección actual, en este caso restaurants, se utilizará para hacer la fusión. El funcionamiento de "foreignField" es muy similar al anterior, pero en este se especifica el campo a utilizar de la colección externa.
Finalmente, tiene el campo "as" donde se especificará el nombre del nuevo campo creado como resultado del lookup. La consulta nos muestra varios resultados como los siguientes.

```
db.restaurants.aggregate([
```

{ "$lookup" : { "from": "inspections", "localField": "_id", "foreignField": "restaurant_id", "as" : "inspection_history" }}])

1. Optimización del rendimiento

En primer lugar, hemos pensado que sería bastante útil una consulta para filtrar por nombre de restaurante y ver el resultado de todas sus inspecciones junto a la fecha en que se hicieron, de esta manera poder realizar estadísticas, según el resultado de las inspecciones, para así detectar posibles problemas en diferentes distritos. La consulta sería la siguiente:

```
db.restaurants.aggregate([
{ "$lookup" : { "from": "inspections", "localField": "_id",
"foreignField": "restaurant_id", "as" : "inspection_history"  }},
{$project: {name: 1, address: 1, "inspection_history.result": 1, "inspection_history.date": 1}} ])
```

```
{
  _id: ObjectId('55f14312c7447c3da7051b84'),
  address: '9 Dromore Road',
  name: '9th Avenue Pizzeria',
  inspection_history: [
    {
      date: 2024-08-20T22:00:00.000Z,
      result: 'Warning Issued'
    },
    {
      date: 2022-06-01T22:00:00.000Z,
      result: 'Warning Issued'
    },
    {
      date: 2024-05-07T22:00:00.000Z,
      result: 'Warning Issued'
    }
  ]
}
```

```
{
  _id: ObjectId('55f14312c7447c3da7051b7b'),
  address: '824 Shields Road',
  name: '7 STAR Pizza',
  inspection_history: [
    {
      date: 2023-07-20T22:00:00.000Z,
      result: 'Fail'
    },
    {
      date: 2024-04-16T22:00:00.000Z,
      result: 'No Violation Issued'
    },
    {
      date: 2023-05-01T22:00:00.000Z,
      result: 'Pass'
    }
  ]
}
```

También, creemos que sería útil una consulta que busque todos aquellos restaurantes que tienen la última inspección fallida, para así saber que restaurantes no se encuentran en las condiciones adecuadas para ofrecer productos a los consumidores y controlar que corrigen aquellos malos hábitos.

```
db.restaurants.aggregate([
 { "$lookup": {  "from": "inspections", "localField": "_id",
    "foreignField": "restaurant_id", "as": "inspection_history" } },
```

```
{ "$addFields": { "last_inspection": { "$arrayElemAt": ["$inspection_history", -1] } } },
{ "$match": { "last_inspection.result": "Fail" } },
{  "$project": { "inspection_history": 0}  } ])
```

En tercer lugar, creamos una consulta que busque aquellos restaurantes con menos inspecciones en los últimos 6 meses. Con esta consulta la auditoría puede planificar y priorizar qué establecimientos deben ser inspeccionados antes que otras que ya han tenido diversas inspecciones.

```
db.inspections.aggregate([
{ "$match": { "date": { "$gte": ISODate("2024-08-01") } } },
{ "$group": { "_id": "$restaurant_id", "total_inspections": { "$sum": 1 },
   "name": { "$first": "$business_name" } } },
{ "$sort": { "total_inspections": 1 } } ])
```

Finalmente, hemos hecho la consulta que busca la última inspección de cada restaurante. Creemos que sería una consulta frecuente para así ver, de manera rápida, en último estado se encontraban los restaurantes. También, permite ver qué restaurantes tienen las medidas sanitarias suficientes y cuales están en peores condiciones.

```
db.restaurants.aggregate ([
 { "$lookup": {  "from": "inspections", "localField": "_id",
    "foreignField": "restaurant_id", "as": "inspection_history"  } },
 {"$addFields": {   "last_inspection": { $arrayElemAt: ["$inspection_history", -1] } } },
 { "$project": { "inspection_history": 0 } } ])
```

```
{
  _id: ObjectId('55f14312c7447c3da7051b89'),
  URL: 'http://www.just-eat.co.uk/restaurants-a-cake-a-shake-stockport/menu',
  address: '8B - 12B Houldsworth Street',
  'address line 2': 'Stockport',
  name: 'A Cake A Shake',
  outcode: 'SK5',
  postcode: '6DA',
  rating: 4.5,
  type_of_food: 'Desserts',
  last_inspection: {
    _id: ObjectId('56d61035a378eccde8a94d5d'),
    id: '19653-2015-ENFO',
    certificate_number: 5377443,
    business_name: 'A CAKE A SHAKE',
    date: 2022-09-16T22:00:00.000Z,
    result: 'Pass',
    sector: 'Cigarette Retail Dealer - 127',
    address: {
      city: 'STOCKPORT',
      zip: '6DA',
      street: '8B - 12B HOULDSWORTH STREET',
      number: ''
    },
    restaurant_id: ObjectId('55f14312c7447c3da7051b89')
  }
}
```