MONGODB

1. En sample_training.zips ¿Cuántas colecciones tienen menos de 1000 personas en el campo pop? (sol. 8065)

db.zips.find({pop:{$lt:1000}}).count()

2. En sample_training.trips ¿Cuál es la diferencia entre la gente que nació en 1998 y la que nació después de 1998? (sol. 6)

```
Math.abs(db.find({"birth year": {$lt:
1998}})).count()-(db.find({"birth year": {$gt:
1998}})).count()
```

3. En sample_training.routes ¿Cuántas rutas tienen al menos una parada? (sol. 11)

```
db.find({"stops": {"$gt": 0 }}).count()
```

4. En sample_training.inspections ¿Cuántos negocios tienen un resultado de inspección "Out of Business" y pertenecen al sector "Home Improvement Contractor - 100"? (sol. 4)

```
db.find({
$and: [
   { "result":{$eq:  "Out of Business"}},
   { "sector": {$eq: "Home Improvement Contractor - 100"}}
]
}).count();
```

```
> db.inspections.find({
  $and: [
     { "result":{$eq:  "Out of Business"}},
     { "sector": {$eq: "Home Improvement Contractor - 100"}}
  ]
  }).count();
< 4
```

ó

db.inspections.find({result:"Out of Business", sector:"Home Improvement Contractor - 100"}).count()

5. En sample_training.inspections ¿Cuántos documentos hay con fecha de inspección "Feb 20 2015" o "Feb 21 2015" y cuyo sector no sea "Cigarette Retail Dealer - 127"? (sol. 204)

```
db.find({ "$and": [
   { "$or" : [ {date: "Feb 20 2015"} , {date: "Feb 21 2015"}]},
   { sector: {"$ne": "Cigarette Retail Dealer - 127"}}
]}).count()
```

Ó

db.inspections.find({$or:[{date:"Feb 20 2015"},{date:"Feb 21 2015"}],sector:{$ne:"Cigarette Retail Dealer - 127"}}).count()

EXPR→ COMPARAR DOS CAMPOS

1. En sample_training.companies, ¿cuántas empresas tienen más empleados que el año en el que se fundaron? (sol. 324)

db.companies.find({$expr:{$gt:["$number_of_employees","$founded_year"]}}).count()

```
> db.companies.find({$expr:{$gt:["$number_of_employees","$founded_year"]}}).count()
< 324
```

2. En sample_training.companies, ¿en cuántas empresas coinciden su permalink con su twitter_username? (sol. 1299)

IGUALAR 2 CAMPOS

```
> db.companies.find({$expr:{$eq:["$permalink","$twitter_username"]}}).count()
< 1299
```

3. En sample_airbnb.listingsAndReviews, ¿cuál es el nombre del alojamiento en el que pueden estar más de 6 personas alojadas y tiene exactamente 50 reviews? (sol. Sunset Beach Lodge Retreat)

CONSULTAS COMPUESTAS

```
db.find({ "$expr":
{"$and":
   [ { "$gt": ["$accommodates", 6]},
     {"$eq": ["$number_of_reviews",
50]}]}},{projection: {_id:0,name:1}}).toArray()
```

Ó

db.listingsAndReviews.find({$and:[{accommodates:{$gt:6}},{number_of_reviews:{$eq:50}}]},{name:1,_id:0})

4.En sample_airbnb.listingsAndReviews, ¿cuántos documentos tienen el "property_type" "House" e incluyen "Changing table" como una de las "amenities"? (sol. 11)

db.listingsAndReviews.find({property_type:"House",amenities:"Changing table"}).count()

```
> db.listingsAndReviews.find({property_type:"House",amenities:"Changing table"}).count()
< 11
```

5. En sample_training.companies, ¿Cuántas empresas tienen oficinas en Seattle? (sol. 117)

db.companies.find({"offices.city":"Seattle"}).count()

db.companies.find({offices:{$elemMatch:{city:"Seattle"}}}).count()

```
> db.companies.find({"offices": {"$elemMatch": {"city": "Seattle" }}}).count()
< 117
```

6. En sample_training.companies, haga una query que devuelve únicamente el nombre de las empresas que tengan exactamente 8 "funding_rounds"

TAMAÑO DE ARRAY CON SIZE

db.companies.find({funding_rounds:{$size:8}},{name:1, _id:0})

```
> db.companies.find({funding_rounds:{$size:8}},{name:1, _id:0})
< {
    name: 'Twitter'
  }
  {
    name: 'LinkedIn'
  }
  {
    name: 'PayScale'
  }
```

7. En sample_training.trips, ¿cuántos viajes empiezan en estaciones que están al oeste de la longitud -74? (sol. 1928)

PARA COMPARAR UNO DE LOS CAMPOS DE UN ARRAY

db.trips.find({'start station location.coordinates.0':{$lt:-74}}).count()

```
db.trips.find({'start station location.coordinates.0':{$lt:-74}}).count()
1928
```

8. En sample_training.inspections, ¿cuántas inspecciones se llevaron a cabo en la ciudad de "NEW YORK"? (sol. 18279)

db.inspections.find({"address.city":"NEW YORK"}).count()

```
> db.inspections.find({ "address.city": "NEW YORK"}).count()
< 18279
```

9. En sample_airbnb.listingsAndReviews, haga una query que devuelva el nombre y la dirección de los alojamientos que tengan "Internet" como primer elemento de "amenities"

TENGAN COMO Nº ELEMENTO EL SIGUIENTE CAMPO

```
sample_airbnb> db.listingsAndReviews.find(
                { "amenities.0": "Internet"},
                { "name": 1 , "address":1, "_id":0})
```

```
sample_airbnb> db.listingsAndReviews.aggregate([
                {"$match":{ "amenities.0": "Internet"}},
               {"$project": {"name": 1 , "address":1, "_id":0}}])
```

1. En sample_airbnb.listingsAndReviews, ¿qué "room types" existen?

TIPOS DE UN CAMPO EXISTENTE

```
> db.listingsAndReviews.aggregate([{ $group: { _id: "$room_type"}}])
< {
    _id: 'Entire home/apt'
  }
  {
    _id: 'Private room'
  }
  {
    _id: 'Shared room'
  }
> db.listingsAndReviews.distinct("room_type")
< [ 'Entire home/apt', 'Private room', 'Shared room' ]
```

2. En sample_training.companies, haga una query que devuelva el nombre y el año en el que se fundaron las 5 compañías más antiguas.

db.companies.find({ founded_year: { $ne: null } }, { _id: 0, name: 1, founded_year: 1 }).sort({ founded_year: 1 }).limit(5)

```
> db.companies.find({ "founded_year": {$ne: null}}, { _id: 0, name: 1, founded_year: 1 }).sort({ founded_year: 1 }).limit(3)
< {
    name: 'Alstrasoft',
    founded_year: 1800
  }
  {
    name: 'US Army',
    founded_year: 1800
  }
  {
    name: 'DuPont',
    founded_year: 1802
  }
```

Ó

```
db.companies.aggregate([
  {
    $match: { "founded_year": { $ne: null } }
  },
  {
    $project: {
      _id: 0,
      name: 1,
      founded_year: 1
    }
  },
  {
    $sort: { "founded_year": 1 }
  },
  {
    $limit: 3
  }
])
```

3. En sample_training.trips, ¿en qué año nació el ciclista más joven? (sol. 1999)

```
db.find({"birth year":{$ne: ""}},{ projection: {_id:0,"birth year":1}}).sort({"birth year": -1}).limit(1).toArray()
```

SIMULACRO

MONGODB

En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más reviews.

ORDENACIÓN POR EL QUE TIENE EL Nº + GRANDE DE UN CAMPO

db.listingsAndReviews.find({}, { _id: 0, name:1, number_of_reviews:1}).sort({number_of_reviews:-1}).limit(1)

```
> db.listingsAndReviews.find({}, { _id: 0, name: 1, number_of_reviews: 1 }).sort({ number_of_reviews: -1 }).limit(1)
< {
    name: '#Private Studio - Waikiki Dream',
    number_of_reviews: 533
  }
```

En la colección listingAndReviews indique el/los nombre(s) del alojamiento con más amenities.

ORDENACIÓN POR EL QUE TIENE EL ARRAY CON MÁS OBJETOS

```
db.aggregate([
  {
    $project: {
      name: 1,
      numAmenities: { $size: "$amenities" }
    }
  },
  {
    $sort: { numAmenities: -1 }
  },
  {
    $limit: 1
  }
```

```
> db.listingsAndReviews.aggregate([
{$project:
{ _id:0 , name: 1, numAmenities: {  $size: "$amenities"}}},
{ $sort: { numAmenities: -1}},
 {$limit: 1
}])
< {
    name: 'Sublime loft with private roof top @ Old Montreal',
    numAmenities: 76
  }
```

En la colección listingAndReviews indique para cada tipo de property_type el número de alojamientos de ese tipo.

```
aggregate([
  {
```

```
    $group: {
      _id: "$property_type",
      count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      property_type: "$_id",
      count: 1
    }
  }
]).toArray()
```

Se agrupan los documentos por el property type, calcula el total de cada grupo y lo almacena en count
$sum→ calcula el total de documentos en cada grupo y lo almacena en count

En la colección listingAndReviews indique el número de alojamientos que tienen 2, 3, 4 o 5 beds

```
db.find({"$or":[
      {"beds":2},
      { "beds":3},
      { "beds":4},
      { "beds":5}]}).count()
```

```
> db.listingsAndReviews.find({"$or":[
        {"beds":2},
        { "beds":3},
        { "beds":4},
        { "beds":5}]}).count()
< 2589
```

ESTA ES LA OTRA OPCIÓN

```
> db.listingsAndReviews.find({ beds: { $in: [2, 3, 4, 5] } }).count()
< 2589
```

ESTA ES LA OTRA OPCIÓN (NO RULA)

```
db.listingAndReviews.aggregate([
  {
    $match: {
      beds: {
        $in: [2, 3, 4, 5]
      }
    }
  },
  {
    $group: {
      _id: "$beds",
      count: { $sum: 1 }
    }
  },
  {
    $project: {
      _id: 0,
      beds: "$_id",
      count: 1
    }
  }
])
```

Con el $match filtra y selecciona solo los que tengan 2,3,4,5 camas

$group agrupamos por beds y calculamos los que hay con $sum:1

Luego projectamos con el project lo que queremos

USO DEL UNWIND→ELIMINA LA DUPLICACIÓN Y FILTRA

Partimos de lo siguiente:

```
{
  "_id": 1,
```

```
  "title": "Libro 1",
  "authors": ["Autor 1", "Autor 2"]
}
{
  "_id": 2,
  "title": "Libro 2",
  "authors": ["Autor 1"]
}
{
  "_id": 3,
  "title": "Libro 3",
  "authors": ["Autor 2", "Autor 3"]
}
```

Realizamos esta consulta

```
db.libros.aggregate([
  {
    $unwind: "$autores"
  },
  {
    $project: {
      _id: 0,
      autor: "$autores"
    }
  }
])
```

NOS DARÁ LA LISTA DE AUTORES DISTINTOS

Resultado:

```
{ autor: "Autor 1" }
{ autor: "Autor 2" }
{ autor: "Autor 3" }
```

BÚSQUEDA DE ARRAY CON ELEMENTOS CONCRETOS

```
Book.find({ author: { $all: [ 'author1', 'author2'], $size: 2 } })
```

esta consulta buscará documentos en la colección "Book" donde el campo "author" contenga tanto "author1" como "author2" y tenga un tamaño de 2 elementos.

Imaginemos ahora que queremos extraer las personas con una edad igual a 25, 30 o 35 años. En SQL podríamos utilizar un *WHERE age IN (25,30,35)*. En **MongoDB** utilizaríamos el operador ***$in*** y un array con los datos.

```
> db.people.find({age:{$in:[25,30,35]}},{name:1,age:1})
```

DONDE EXISTA UN CAMPO

```
> db.people.find({company:{$exists:true}},{name:1,age:1,company:1
```

DISTINCT PARA LISTADO DE LOS DIFERENTES BEDS QUE HAY(COMO EL UNWIND)

```
> db.listingsAndReviews.distinct("beds")
< [
    0,  1,  2,  3,  4,  5,  6,
    7,  8,  9, 10, 11, 12, 13,
   14, 15, 16, 18, 25
  ]
```

EQUIVALENTE A ESTA CON AGGREGATE

db.listingsAndReviews.aggregate([{ $group: { _id: "$beds"}}])

EXAMEN FINAL

1.En la colección movies, indique el titulo y el nº de premios de la pelicula con mas premios (awards.wins)

```
db.movies.aggregate([
  {
    $project: {
      title: 1,
      numAwards: { $toInt: "$awards.wins" }
    }
  },
  {
    $sort: { numAwards: -1 }
  },
  {
    $limit: 1
  }
])
```

2. En la colección movies muestra un listado con las diferentes clasificaciones de edad que existen (rated) Para cada uno de ellos muestre el nº de documentos que tiene esa clasificación

GROUP PARA AGRUPAR POR UN CAMPO Y CONTAR CUÁNTOS DOCUMENTOS HAY

```
db.movies.aggregate([
  {
    $group: {
      _id: "$rated",
      count: { $sum: 1 }
    }
  }
])
```

3.En la colección movies muestre un listado con los diferentes generos de peliculas que existen (genres)

```
db.movies.distinct("genres")
```

Ó

```
db.movies.aggregate([
  {
```

```
    $unwind: "$genres"
  },
  {
    $group: {
      _id: null,
      genres: { $addToSet: "$genres" }
(addToSet→para crear un arreglo único de géneros sin duplicados.)
    }
  },
  {
    $project: {
      _id: 0,
      genres: 1
    }
  }
])
```

CONSULTAS Listing And Reviews:

https://www.w3resource.com/mongodb-exercises/mongodb-listingsandreviews-collection-index.php

https://gist.github.com/iamramann/4838049b3faeb37e45482fed0a11aa61

https://www.mongodb.com/docs/manual/reference/operator/aggregation/group/

Repo práctica:

https://github.com/mikimb99/Proyecto-SWII.git

Repo Alvaro:

https://bitbucket.org/alvarosp_ceu/sw2_2023_repository/src/master/simulacro/api/routes/

mongosh

use database

db.collection.find() acuérdate de pegarla así en el archivo!

OPENAPI

COMPROBADOR DE OPENAPI:

https://editor.swagger.io/

EJEMPLO BÁSICO OPENAPI
https://github.com/aluna1997/ApiMoonTravelsSwagger/blob/main/Clientes%20Swagger.yaml+

Las respuestas tienen que ser en base a los códigos

```
paths:
    /board:
        get:
            summary: Get the whole board
            description: Retrieves the current state of the board and the
        winner.
            responses: IMPORTANTE LAS QUE TE PIDAN EN LOS MÉTODOS

            "200": description: "OK"
             content:
                application/json:
                    schema:
                        type: string
                        …
```

/users/{id}:	/users
…	…
-name: id	-name: id
in: path	in: query

Cuando son post o put añadir el requestBody:

```
post:
summary: Crear nueva película
description: TODO
requestBody:
required: true
```

```
content:
application/json:
schema:
$ref: "#/components/schemas/pelicula"
```

Cuando aparte de la respuesta vamos a regresar info ponemos content:

```
responses:
"201":
description: "Created"
content:
application/json:
schema:
$ref: "#/components/schemas/nuevoElemento"
```

DENTRO DE SCHEMAS QUEDARÍA ASÍ LA RESPUESTA

```
nuevoElemento:
type: object
properties:
id:
$ref: "#/components/schemas/id"
url:
description: Enlace a la sesión o película creada
type: string
format: uri
example: {"id": 123,
"url": "http://example.com/api/v1/peliculas/123"}
```

Cuando es un path con un atributo dentro:

```
/peliculas/{id}:
parameters:
- $ref: "#/components/parameters/id"
get:
summary: Obtener información de una película concreta
description: TODO
```

Recordar que hay que añadir parameters, en este caso lo referencio en parameters y quedaría así:

```yaml
components:
parameters:
id:
description: id de una película o sesión
name: id
in: path
required: true
schema:
$ref: "#/components/schemas/id"
```

El $ref de schemas simplemente sería:

```yaml
id:
type: integer
minimum: 1
example: 123
```

Podrías hacerlo todo directamente sin los $ref de esta forma:

```yaml
paths:
  /users/{id}:
    get:
      parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
```

$ref al final es útil cuando necesitas referenciar a varios

UTILIZACIÓN DEL ALLOF

```yaml
BookMin:
  type: object
  properties:
    _id:
      $ref: "#/components/schemas/ID"
    title:
      type: string
      description: Book title
    author:
      type: string
      description: Book author
  required:
    - _id
    - title
    - author
```

```yaml
Book:
  allOf:
    - $ref: "#/components/schemas/BookMin"
  properties:
    link:
      type: string
      description: URL to the book's details
  required:
    - link
```

Como en este caso ya había definido previamente los required para un libro y solo quiero añadir el link, solo necesito tomar ref de todo el Bookmin y añadir el link