Sistemas Web II: MongoDB

Ejercicio 1

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();

```
sampleDB> db.zips.findOne()
{
  _id: ObjectId('5c8eccc1caa187d17ca6ed16'),
  city: 'ALPINE',
  zip: '35014',
  loc: { y: 33.331165, x: 86.208934 },
  pop: 3062,
  state: 'AL'
}
sampleDB> db.zips.find({"pop":{"$lt":1000}}).count()
8065
```

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)

(db.trips.find({ 'birth year': { $gt: 1998 }}).count())-(db.trips.find({ 'birth year': 1998 }).count());

```
sampleDB> db.trips.find({ 'birth year': { $gt: 1998 }}).count();
18
sampleDB> db.trips.find({ 'birth year': { $gt: 1998 }}).count();
sampleDB> db.trips.find({ 'birth year': 1998 }).count();
12
sampleDB> (db.trips.find({ 'birth year': { $gt: 1998 }}).count())-( db.trips.find({ 'birth year': 1998 }).count());
6
sampleDB>
```

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

db.routes.countDocuments({ "stops": { $gte: 1 } });

```
sampleDB> db.routes.findOne();
{
  _id: ObjectId('56e9b39b732b6122f877fa31'),
  airline: { id: 410, name: 'Aerocondor', alias: '2B', iata: 'ARD' },
  src_airport: 'CEK',
  dst_airport: 'KZN',
  codeshare: '',
  stops: 0,
  airplane: 'CR2'
}
sampleDB> db.routes.countDocuments({ "stops": { $gte: 1 } });
11
sampleDB>
```

4. En sample_training.inspections, ¿Cuántos negocios tienen un resultado de inspección "OutofBusiness" y pertenecen al sector "Home ImprovementContractor-100"? (sol. 4)

Listar Valores Únicos del Campo ‘result’:

```
sampleDB> db.inspections.distinct("result");
[
  'Business Padlocked',
  'Closed',
  'Completed',
  'Condemned',
  'Confiscated',
  'ECB Summons Issued',
  'ECB Warning Issued',
  'Fail',
  'License Confiscated',
  'Licensed',
  'NOH Withdrawn',
  'No Evidence of Activity',
  'No Violation Issued',
  'Out of Business',
  'Pass',
  'Posting Order Served',
  'Re-inspection',
  'Samples Obtained',
  'Unable to Complete Inspection',
  'Unable to Locate',
  'Unable to Seize Vehicle',
  'Violation Issued',
  'Warning'
]
sampleDB>
```

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

```
sampleDB> db.inspections.countDocuments({ result: "Out of Business", sector: "Home Improvement Contractor - 100" });
4
sampleDB>
```

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

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

```
sampleDB> db.inspections.countDocuments({  date: { $in: ["Feb 20 2015", "Feb 21 2015"] },  sector: { $ne: "Cigarette Retail Dealer - 127" } });
204
sampleDB>
```

Ejercicio 2

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

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

```
sampleDB> db.companies.countDocuments({
...       $expr: {
...   $gt: ["$number_of_employees", "$founded_year"]
...       }
... });
324
sampleDB> db.companies.countDocuments({ $expr: { $gt: ["$number_of_employees", "$founded_year"] } });
324
sampleDB>
```

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

db.companies.countDocuments({ $expr: { $eq: ["$permalink", "$twitter_username"] } });

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. SunsetBeach LodgeRetreat)

db.listings.find({ accommodates: { $gt: 6 }, number_of_reviews: 50 }, { name: 1, _id: 0 });
ó
db.listingsAndReviews.find({accommodates: {$gt: 6}, number_of_reviews: {$eq: 50}}, {_id: 0, name: 1})

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

```
db.listings.countDocuments({  property_type: "House",  amenities: "Changing table"});
```

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

```
db.companies.countDocuments({  "offices.city": "Seattle" });
```

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

Nota: devolver únicamente el nombre de las empresas que tienen exactamente 8 elementos en el array funding_rounds

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

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

Nota 1: Hacia el oeste la longitud decrece
Nota 2: el formato es <field_name>: [<longitud>, <latitud>]

```
db.trips.countDocuments({  'start station location.coordinates.0': { $lt: -74 } });
```

Explicación de la Consulta:

- Filtro (query):
  'start station location.coordinates.0': { $lt: -74 }: Selecciona documentos donde la longitud de la ubicación de la estación de inicio (primer valor en el array coordinates) es menor que -74. Esto corresponde a ubicaciones al oeste de la longitud -74.

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

```
db.inspections.countDocuments({  "address.city": "NEW YORK" });
```

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

```
db.listings.find({  "amenities.0": "Internet"}, {  "name": 1,  "address": 1,  "_id": 0 }).pretty();
```

Ejercicio 3

1. En sample_airbnb.listingsAndReviews, ¿qué "roomtypes" existen?

db.listings.aggregate([{ $group: { _id: "$room_type" } }, { $project: { _id: 0, room_type: "$_id" } }])

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.

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

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

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

Simulacro

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

```
db.listings.aggregate([
  { $sort: { number_of_reviews: -1 } },
  { $limit: 1 },
  { $project: { _id: 0, name: 1, number_of_reviews: 1 } }
])
```

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

```
db.listings.aggregate([
  { $project: { name: 1, number_of_amenities: { $size: "$amenities" } } },
  { $sort: { number_of_amenities: -1 } },
  { $limit: 1 },
  { $project: { _id: 0, name: 1, number_of_amenities: 1 } }
])
```

ó

```
db.listings.sort({“amenities.length”:-1}).limit(1)
```

ó

```
db.listings.find({$expr: { $max: [{$size: “$amenities”}]}}).limit(1)
```

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

```
db.listings.aggregate([
  { $group: { _id: "$property_type", count: { $sum: 1 } } },
  { $project: { _id: 0, property_type: "$_id", count: 1 } },
  { $sort: { count: -1 } }
])
```

ó

```
db.listingsAndReviews.aggregate({$group: {id:”$property_type”,”count”: {$sum:1}}})
```

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

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

ó

```
db.listingsAndReviews.find({$and: [ {bed: {$gte: 2}, {bed: {$lte: 5}} } ] } ).count()
```

ó

```
db.listingsAndReviews.countDocuments({beds:{$in:[2,3,4,5] } })
```

Expressive, Arrays, Projection, Aggregation, Cursor methods

Expressive $expr

El operador $expr en MongoDB permite utilizar expresiones de agregación en las consultas de búsqueda (find). Esto permite realizar comparaciones y cálculos entre los campos dentro del mismo documento.

Ej. 1: Número de documentos de sample_training.trips donde el viaje empieza y termina en la misma estación:

```
db.trips.find({ "$expr": { "$eq": [ "$end station id", "$start station id"] }}).count()
```

Explicación:

1. $expr: Este operador permite usar expresiones de agregación para evaluar condiciones.
2. **$eq**: Es un operador de comparación que **verifica si los dos valores son iguales**.
3. ["$end station id", "$start station id"]: **Compara los valores de los campos** $end station id y $start station id en cada documento.

Ej. 2: Find all documents where the trip lasted longer than 1200 seconds, and started and ended at the same station:

```
db.trips.find({ "$expr":
        { "$and": [
        { "$gt": [ "$tripduration", 1200 ]},
        { "$eq": [ "$endstationid", "$startstationid" ]}
]}}).count()
```

Explicación:

1. $expr: Utiliza expresiones de agregación para evaluar las condiciones.
2. $and: Es un operador lógico que evalúa múltiples expresiones y devuelve true sólo si todas las expresiones son true.
3. $gt: Es un operador de comparación que verifica si el primer valor es mayor que el segundo.
4. ["$tripduration", 1200]: Verifica si el valor del campo tripduration es mayor que 1200 segundos.
5. $eq: Es un operador de comparación que verifica si los dos valores son iguales.
6. ["$endstationid", "$startstationid"]: Verifica si el valor del campo endstationid es igual al valor del campo startstationid.

Arrays

"$all": […]

- Para buscar arrays que contengan **al menos esos elementos**

db.listingsAndReviews.find({ "amenities": { **"$all"**: ["Internet", "Wifi", "Kitchen", "Heating", "Family/kidfriendly", "Washer", "Dryer", "Essentials"] }})

"$size": number

- Para especificar un **tamaño exacto del array**

db.listingsAndReviews.find({"amenities":{**"$size"**:55}}).count()

Recordatorio, en los arrays importa el orden:

{"amenities": "Shampoo"}

- Permite buscar en el array

{"amenities": ["Shampoo"]}

- Busca exactamente ese array

Projection

Permite obtener sólo los campos que queremos. Proceso de especificar qué campos deben ser incluidos o excluidos en los documentos devueltos por una consulta (útil cuando solo necesitas una parte de los datos almacenados en los documentos de la colección, puede mejorar el rendimiento de la consulta al reducir la cantidad de datos transferidos).

Sintaxis Básica:

La proyección se especifica como el segundo argumento del método find:

db.collection.**find**(query, projection)

- {<field1>: 1, <field2>:1,…}
  Se incluyen esos campos (además de _id), se excluyen el resto

- {<field1>: 0, <field2>: 0,…}
  Se excluyen esos campos, se incluyen todos los demás

- No se pueden mezclar 0s y 1s excepto para excluir _id

EJEMPLO PROJECTION

```
db.listingsAndReviews.find({ "amenities": "Wifi" }, { "price": 1, "address": 1})
```

ó

```
db.listingsAndReviews.find({ "amenities": "Wifi" }, { "price": 1, "address": 1, "_id": 0 })
```

NO: `db.listingsAndReviews.find({ "amenities": "Wifi" }, { "price": 1, "address": 1, "maximum_nights":0 })`. En MongoDB, cuando utilizas una proyección para incluir ciertos campos (especificando 1), debes ser consistente y solo usar 0 para excluir campos adicionales. No puedes mezclar inclusiones y exclusiones (excepto con el campo _id).

Desglose de la Consulta: Busca documentos que contienen "Wifi" en el campo amenities y proyecta solo los campos price y address en los resultados.

- Filtro (query): { "amenities": "Wifi" }
  - Esta parte de la consulta busca documentos donde el campo amenities (que se espera sea un array) contiene el valor "Wifi".
- Proyección (projection): { "price": 1, "address": 1, _id: 0 }
  - "price": 1: Incluye el campo price en los documentos devueltos.
  - "address": 1: Incluye el campo address en los documentos devueltos.
  - "_id": 0: Excluye el campo _id de los documentos devueltos (opcional, pero a menudo útil para reducir el volumen de datos si el _id no es necesario).

Supongamos que la colección listingsAndReviews contiene documentos con la siguiente estructura:

```
{
  "_id": "10006546",
  "listing_url": "https://www.airbnb.com/rooms/10006546",
  "amenities": ["Wifi", "Kitchen", "Heating", ...],
  "price": "$85.00",
  "address": {
    "street": "123 Main St",
    "city": "San Francisco",
    "country": "United States",
    ...
  },
  ...
}
```

La consulta devolverá documentos que contienen solo los campos price y address para aquellos listados que tienen "Wifi" en sus amenities:

```
[
  {
    "price": "$85.00",
    "address": {
      "street": "123 Main St",
      "city": "San Francisco",
      "country": "United States"
    }
  }
```

Projection

"$elemMatch"

- Usado con arrays, los proyecta sólo si tienen un elemento que cumpla el criterio

```
db.grades.find({"class_id": 431}, {"scores": {"$elemMatch": {"score": {"$gt": 85}}}})
```

"<field1>.<field2>.<field3>…"

- **Dotnotation**
- Usada para recorrer subdocumentos
- Se pueden usar números para las posiciones de los arrays
- Tiene que estar entre comillas

```
db.companies.find( { "relationships.0.person.last_name": "Zuckerberg"}, { "name": 1 })
```

"$regex"

- Para buscar patrones en Strings usando expresiones regulares

```
db.companies.find({ "relationships.0.person.first_name": "Mark", "relationships.0.title": {"$regex": "CEO" } }, { "name": 1 })
```

Aggregation

Operación que permite procesar y transformar documentos en una colección. Se utiliza para realizar operaciones de agrupamiento, filtrado, ordenación y transformación de datos, entre otras. La agregación se lleva a cabo principalmente mediante el método aggregate, que acepta una lista de etapas del pipeline.

Conceptos Clave

- **Pipeline**: Una secuencia de etapas a través de las cuales pasan los documentos. Cada etapa transforma los documentos y los pasa a la siguiente etapa.
- **Stage** (Etapa): Cada etapa del pipeline aplica una operación a los documentos. Algunas etapas comunes incluyen $match, $group, $project, $sort, $limit, $skip, etc.

1. Básico: Contar documentos agrupados por un campo específico

```
db.inspections.aggregate([  {  $group: {   _id: "$address.city",  count: { $sum: 1 }   } ]);
```

2. Filtrar y agrupar:

```
db.inspections.aggregate([  { $match: { result: "No Violation Issued" }  },
                            {   $group: {  _id: "$address.city",   count: { $sum: 1 }  } }]);
```

3. Proyectar campos específicos:

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

4. Ordenar y limitar resultados:

```
db.listings.aggregate([
                    {  $match: { amenities: "Wifi" }  },
                    {  $sort: { price: -1 } },
```

```
  {  $limit: 5  },
  {  $project: {  _id: 0,name: 1, price: 1, address: 1 } }]);
```

Cursor methods

Métodos para realizar consultas, actualizaciones, eliminaciones y otras operaciones en las colecciones. Se aplican a un cursor (después de la consulta, por ejemplo después de que devuelva un find().

count(), limit(), pretty(), skip(), sort(), find(), findOne(), countDocuments(),