T4: EJ 2 MongoDB

lunes, 12 de mayo de 2025 17:48

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> ]`

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

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

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

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"`

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

Apartado 1.

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

Apartado 2.

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

Apartado 3.

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

Apartado 4.

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

1. Mostrar los 10 primeros alojamientos

- Colección: `sample_airbnb.listingsAndReviews`

1. db.listingsAndReviews.find({},{"name":1, "_id":0}).limit(10)

2. Listar los alojamientos en "Spain" cuyo precio sea menor de 100€

- Colección: `sample_airbnb.listingsAndReviews`

2. db.listingsAndReviews.find({"$and": [{"price": {"$lt": 100}},{"address.country": "Spain"}]})

3. Contar el número de alojamientos por "room_type"

- Colección: `sample_airbnb.listingsAndReviews`

3. db.listingsAndReviews.aggregate([
{
$group: {
_id: "$room_type",
count: { $sum: 1 }
}
},
{
$project: {
_id: 0,
room_type: "$_id",
count: 1
}
}
])

3'. sample_airbnb> db.listingsAndReviews.aggregate([{"$group": {"_id": "$room_type"}}])
3''. db.listingsAndReviews.distinct("room_type")

4. Obtener el precio medio (average price) por "property_type"

- Colección: `sample_airbnb.listingsAndReviews`

db.listingsAndReviews.aggregate([{ $group: { _id: "$property_type", averagePrice: { $avg: "$price" } } }, { $project: { _id: 0, property_type: "$_id", averagePrice: 1 } }, { $sort: { averagePrice: -1 } }])

5. Encontrar los alojamientos que tengan "WiFi" y "Air conditioning" en la lista de amenities

- Colección: `sample_airbnb.listingsAndReviews`

5. sample_airbnb> db.listingsAndReviews.find({"amenities": {"$all": ["Wifi", "Air conditioning"]}},{"name": 1, "_id": 0})

6. Mostrar los 5 alojamientos con mejor puntuación en "review_scores.review_scores_rating"

- Colección: `sample_airbnb.listingsAndReviews`

db.listingsAndReviews.find({ "review_scores.review_scores_rating": { $exists: true } }, { name: 1, "review_scores.review_scores_rating": 1, "address.city": 1, _id: 0 }) .sort({ "review_scores.review_scores_rating": -1 }) .limit(5) .pretty()

```
# Soluciones al problema 2

## Apartado 1

db.listingsAndReviews.aggregate([ { "$group": { "_id": "$name", "ReviewsDisponibles": { "$sum": {"$size": "$reviews" }}}}]).sort
({"ReviewsDisponibles": -1})

sw2> db.listingsAndReviews.aggregate([ { "$group": { "_id": "$name", "ReviewsDisponibles": { "$sum": {"$size": "$reviews" }}}},
{"$sort":{"ReviewsDisponibles": -1}}, {"$limit": 1}])

db.listingsAndReviews.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      numReviews: { $size: "$reviews" }
    }
  },
  { $sort: { numReviews: -1 } },
  { $limit: 1 }
])

## Apartado 2

sw2> db.listingsAndReviews.aggregate([ { "$group": { "_id": "$name", "AmenitiesDisponibles": { "$sum": {"$size": "$amenities"
}}}}]).sort({"AmenitiesDisponibles": -1})

sw2> db.listingsAndReviews.aggregate([ { "$group": { "_id": "$name", "AmenitiesDisponibles": { "$sum": {"$size": "$amenities" }}}},
{"$sort":{"AmenitiesDisponibles": -1}}, {"$limit": 1}])


## Apartado 3

sw2> db.listingsAndReviews.aggregate([ { "$group": { "_id": "$property_type", "totalAlojamientos": { "$sum": 1 } } } ] )

## Apartado 4

sw2> db.listingsAndReviews.find({"$expr": {"$or": [{"$gte": ["$beds", 2]}, {"$lte": ["$beds", 5]}]}}).count()
```

7. Listar los nombres de los hosts (sin duplicados) que tengan más de 3 alojamientos publicados

- Colección: `sample_airbnb.listingsAndReviews`

db.listingsAndReviews.aggregate([{ $group: { _id: "$host.host_name", totalListings: { $sum: 1 } } }, { $match: { totalListings: { $gt: 3 } } }, { $project: { _id: 0, host_name: "$_id", totalListings: 1 } }])

8. Buscar alojamientos en un rango geográfico concreto

- Colección: `sample_airbnb.listingsAndReviews`
- Campo: `address.location.coordinates`, donde cada documento almacena `[longitud, latitud]`. Define dos puntos (por ejemplo, suroeste y noreste de un rectángulo) y busca dentro de ese "box" geográfico.

```
db.listingsAndReviews.createIndex({ "address.location": "2dsphere" })
```

2. **Consulta dentro de un rectángulo** (ejemplo aproximado para Barcelona):

```
db.listingsAndReviews.find(
  {
    "address.location": {
      $geoWithin: {
        $box: [
          [2.10, 41.35], // suroeste (longitud, latitud)
          [2.20, 41.45]  // noreste (longitud, latitud)
        ]
      }
    }
  },
  {
    name: 1,
    "address.city": 1,
    "address.location": 1,
    _id: 0
  }
).pretty()
```

10. Buscar alojamientos cuyo nombre contenga la palabra "Loft" (independientemente de mayúsculas/minúsculas)

```
db.listingsAndReviews.find(
  { name: { $regex: /Loft/i } },
  { name: 1, "address.city": 1, price: 1, _id: 0 }
).pretty()
```

11. Calcular el número total de reviews por mes

- Colección: `sample_airbnb.listingsAndReviews`
- Campo: `reviews`, que es un arreglo donde cada elemento tiene un campo `date` (fecha de la reseña). Debes "desplegar" ese arreglo (uno por uno) y luego extraer año y mes de cada fecha para agrupar.

```
db.listingsAndReviews.aggregate([
  // Desenrollar el array de 'reviews'
  { $unwind: "$reviews" },
  // Extraer año-mes de cada fecha de reseña
  {
    $project: {
      year_month: {
        $dateToString: { format: "%Y-%m", date: "$reviews.date" }
      }
    }
  },
  // Agrupar por año-mes y contar reseñas
  {
    $group: {
      _id: "$year_month",
      total_reviews: { $sum: 1 }
    }
  },
  // Proyectar campos finales
  {
    $project: {
      _id: 0,
      year_month: "$_id",
      total_reviews: 1
    }
  },
  // Ordenar cronológicamente
  {
    $sort: { year_month: 1 }
  }
])
```

```
db.listingsAndReviews.aggregate([
  {
    $match: {
      "review_scores.review_scores_rating": { $exists: true }
    }
  },
  {
    $group: {
      _id: "$address.market",
      ratingPromedio: { $avg: "$review_scores.review_scores_rating" }
    }
  },
  {
    $project: {
      _id: 0,
      barrio: "$_id",
      ratingPromedio: 1
    }
  },
  {
    $sort: { ratingPromedio: -1 }
  }
])
```

2. Uso de `$sum`

Objetivo: Calcular el número total de camas (`beds`) por cada barrio (`address.market`).

js

```
db.listingsAndReviews.aggregate([
  {
    $group: {
      _id: "$address.market",
      totalCamas: { $sum: "$beds" }
    }
  },
  {
    $project: {
      _id: 0,
      barrio: "$_id",
      totalCamas: 1
    }
  },
  {
    $sort: { totalCamas: -1 }
  }
])
```

9. Proyección de campos anidados: obtener sólo nombre, barrio ("address.market") y número de camas

- Colección: `sample_airbnb.listingsAndReviews`
- Campos: `name`, `address.market` (o lo que internamente se use para "barrio/vecindario"), y `beds`. Omite el resto.

db.listingsAndReviews.find({}, { _id: 0, name: 1, "address.market": 1, beds: 1 }).pretty()

```
db.listingsAndReviews.aggregate([
  {
    $group: {
      _id: "$room_type",
      precioPromedio: { $avg: "$price" },
      precioMin: { $min: "$price" },
      precioMax: { $max: "$price" }
    }
  },
  {
    $project: {
      _id: 0,
      tipoHabitación: "$_id",
      precioPromedio: 1,
      precioMínimo: "$precioMin",
      precioMáximo: "$precioMax"
    }
  },
  {
    $sort: { precioPromedio: -1 }
  }
])
```

1. Uso de `$count`

Objetivo: Contar cuántos alojamientos incluyen "Breakfast" en sus amenities.

js

```
db.listingsAndReviews.aggregate([
  {
    $match: {
      amenities: { $in: ["Breakfast"] }
    }
  },
  {
    $count: "totalConBreakfast"
  }
])
```

3. Uso de `$first` y `$last`

Objetivo: Para cada tipo de propiedad (`property_type`), obtener el nombre del alojamiento más barato y el nombre del alojamiento más caro.

Nota importante: Para que `$first` y `$last` funcionen correctamente, primero debes ordenar los documentos en el pipeline según el campo que te interesa (aquí `price`).

```
db.listingsAndReviews.aggregate([
  // 1. Ordenar globalmente por property_type y luego por price ascendente
  {
    $sort: {
      property_type: 1,
      price: 1
    }
  },
  // 2. Agrupar por cada property_type
  {
    $group: {
      _id: "$property_type",
      alojamientoMásBarato: { $first: "$name" },
      precioMásBarato: { $first: "$price" },
      alojamientoMásCaro: { $last: "$name" },
```

```
    }
  },
  {
    $project: {
      _id: 0,
      barrio: "$_id",
      totalCamas: 1
    }
  },
  {
    $sort: { totalCamas: -1 }
  }
])
```

```
db.listingsAndReviews.aggregate([
  // 1. Ordenar globalmente por property_type y luego por price ascendente
  {
    $sort: {
      property_type: 1,
      price: 1
    }
  },
  // 2. Agrupar por cada property_type
  {
    $group: {
      _id: "$property_type",
      alojamientoMásBarato: { $first: "$name" },
      precioMásBarato: { $first: "$price" },
      alojamientoMásCaro: { $last: "$name" },
      precioMásCaro: { $last: "$price" }
    }
  },
  // 3. Proyectar campos finales con nombres legibles
  {
    $project: {
      _id: 0,
      tipoPropiedad: "$_id",
      alojamientoMásBarato: 1,
      precioMásBarato: 1,
      alojamientoMásCaro: 1,
      precioMásCaro: 1
    }
  },
  // 4. Orden opcional por tipo de propiedad
  {
    $sort: { tipoPropiedad: 1 }
  }
])
```

El operador `$in` comprueba si al menos uno de los valores está presente en el array, no todos. Por ejemplo:

js Copiar Editar

```
// Esto devuelve alojamientos que tengan "Breakfast" o "Wifi" (o ambos),
// porque con $in basta que coincida uno solo:
db.listingsAndReviews.aggregate([
  {
    $match: {
      amenities: { $in: ["Breakfast", "Wifi"] }
    }
  },
  {
    $count: "totalConBreakfastOWifi"
  }
])
```

Si lo que buscas es que el campo `amenities` contenga **ambos** valores ("Breakfast" y "Wifi"), debes usar `$all` en lugar de `$in`:

js Copiar Editar

```
db.listingsAndReviews.aggregate([
  {
    $match: {
      amenities: { $all: ["Breakfast", "Wifi"] }
    }
  },
  {
    $count: "totalConBreakfastYWifi"
  }
])
```

- `$in: ["Breakfast", "Wifi"]` → verdadero si al menos uno de los dos está en el array.
- `$all: ["Breakfast", "Wifi"]` → verdadero solo si ambos aparecen en el array.