

MONGODB

Preparación de la base de datos:

Aunque la base de datos se encuentra creada desde el ejercicio de repaso, se creará tanto una copia de seguridad de la colección completa como una colección de prueba mediante el código aportado por el enunciado.

Copia de la colección:

```
> show collections
backup_datos_sensores
datos_sensores
> db.backup_datos_sensores.count()
5375
> db.datos_sensores.count()
5375
> db.datos_sensores.aggregate([{$match:{}}, {$out: "backup_datos_sensores"}])
```

Colección de “prueba” conteniendo los 8 primeros documentos de la colección:

```
BulkWriteResult({
    "writeErrors" : [ ],
    "writeConcernErrors" : [ ],
    "nInserted" : 8,
    "nUpserted" : 0,
    "nMatched" : 0,
    "nModified" : 0,
    "nRemoved" : 0,
    "upserted" : [ ]
})
> show collections
backup_datos_sensores
datos_sensores
test
```

Con una copia de seguridad y una pequeña colección para realizar los test de código, podemos comenzar a resolver los casos.

NOTA: los códigos, una vez probados en db.test, serán formateados en un IDE y copiados en la consola de Linux para facilitar su lectura.

CASO 1: crear una nueva colección llamada “sensores” e introducir en ella cuatro documentos, correspondientes a cada uno de los sensores que se encuentran instalados en este momento: dos en Sevilla y dos en Valladolid. La colección deberá contener las siguientes claves:

- Ubicacion: tipo *string*, indica la ciudad en la que está instalado el sensor.
- “medidas_sensor”: *array* de objetos JSON. Cada JSON tendrá dos subclaves: “tipo_medida”, *string* que identifica lo que mide el sensor; “unidad”: *string* que identifica las unidades en las que mide el sensor cada tipo de medida.
- Coordenadas: *array* de numéricos. Los sensores de Valladolid tienen las coordenadas [41.638597, -4.740186] y los de Sevilla [37.409311, -5.949939].
- “fecha_instalación”: *timestamp*. Los dos sensores de Valladolid se instalaron el día 25 de mayo 2020 a las 10:30 AM, mientras que los de Sevilla se instalaron el 28 de mayo 2020 a las 11:30 AM.
- “location_id”: numérico que indica el ID de la ubicación a la que pertenece el sensor. Valor uno para los sensores de Valladolid y valor dos para los sensores de Sevilla.
- “tipo_sensor”: numérico que indica el tipo de sensor. Valor uno para el tipo de sensor que mide temperatura y humedad relativa. Valor dos para el tipo de sensor que mide emisión de CO2 y consumo eléctrico.

La colección se creará automáticamente al insertar los cuatro documentos en una misma consulta. Adjuntar captura de la consulta de inserción, así como el resultado tras ejecutarla.

```
> db.sensores.insertMany([
...   {
...     "Ubicacion": "Valladolid",
...     "medidas_sensor": [
...       { "tipo_medida": "Temperatura", "unidad": "°C" },
...       { "tipo_medida": "Humedad_relativa", "unidad": "%" }
...     ],
...     "Coordenadas": [41.638597, -4.740186],
...     "fecha_instalacion": ISODate("2020-05-25T10:30:00Z"),
...     "location_id": 1,
...     "tipo_sensor": 1
...   },
...   {
...     "Ubicacion": "Valladolid",
...     "medidas_sensor": [
...       { "tipo_medida": "Emision_CO2", "unidad": "gCO2/m2" },
...       { "tipo_medida": "Consumo_electrico", "unidad": "kWh/m2" }
...     ],
...     "Coordenadas": [41.638597, -4.740186],
...     "fecha_instalacion": ISODate("2020-05-25T10:30:00Z"),
...     "location_id": 1,
...     "tipo_sensor": 2
...   },
...   {
...     "Ubicacion": "Sevilla",
...     "medidas_sensor": [
...       { "tipo_medida": "Temperatura", "unidad": "°C" },
...       { "tipo_medida": "Humedad_relativa", "unidad": "%" }
...     ],
...     "Coordenadas": [37.409311, -5.949939],
...     "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
...     "location_id": 2,
...     "tipo_sensor": 1
...   },
...   {
...     "Ubicacion": "Sevilla",
...     "medidas_sensor": [
...       { "tipo_medida": "Emision_CO2", "unidad": "gCO2/m2" },
...       { "tipo_medida": "Consumo_electrico", "unidad": "kWh/m2" }
...     ],
...     "Coordenadas": [37.409311, -5.949939],
...     "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
...     "location_id": 2,
...     "tipo_sensor": 2
...   }
])
```

```

...
{
    "Ubicacion": "Sevilla",
    "medidas_sensor": [
        { "tipo_medida": "Temperatura", "unidad": "°C" },
        { "tipo_medida": "Humedad_relativa", "unidad": "%" }
    ],
    "Coordenadas": [37.409311, -5.949939],
    "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
    "location_id": 2,
    "tipo_sensor": 1
},
{
    "Ubicacion": "Sevilla",
    "medidas_sensor": [
        { "tipo_medida": "Emision_CO2", "unidad": "gCO2/m2" },
        { "tipo_medida": "Consumo_electrico", "unidad": "kWh/m2" }
    ],
    "Coordenadas": [37.409311, -5.949939],
    "fecha_instalacion": ISODate("2020-05-28T11:30:00Z"),
    "location_id": 2,
    "tipo_sensor": 2
}
];

```

Resultado:

```

> show collections
backup_datos_sensores
datos_sensores
sensores
test
> db.sensores.count()
4

```

```

> db.sensores.findOne({})
{
    "_id" : ObjectId("66f83364d0a1f21f095fcb2e"),
    "Ubicacion" : "Valladolid",
    "medidas_sensor" : [
        {
            "tipo_medida" : "Temperatura",
            "unidad" : "°C"
        },
        {
            "tipo_medida" : "Humedad_relativa",
            "unidad" : "%"
        }
    ],
    "Coordenadas" : [
        41.638597,
        -4.740186
    ],
    "fecha_instalacion" : ISODate("2020-05-25T10:30:00Z"),
    "location_id" : 1,
    "tipo_sensor" : 1
}

```

CASO 2: mostrar el número de datos recogidos por el sensor que mide la temperatura en Valladolid, así como el número de datos recogidos por el de Sevilla entre los días 1 y 10 julio (incluido el día 10 entero).

Haciendo cuentas, se reciben cuatro datos por hora, 96 por día, y, por tanto, en 10 días debería haber 960 datos enviados por cada sensor. Comentar si no se han recibido datos de temperatura en algún intervalo de 15 minutos para alguno de los sensores.

```
> db.datos_sensores.count({  
...   "location_id": 2,  
...   "sensor_id": 1,  
...   "timestamp": {  
...     $gte: "2020-07-01T00:00:00Z",  
...     $lte: "2020-07-10T23:59:59Z"  
...   }  
... })  
959  
> db.datos_sensores.count({  
...   "location_id": 1,  
...   "sensor_id": 1,  
...   "timestamp": {  
...     $gte: "2020-07-01T00:00:00Z",  
...     $lte: "2020-07-10T23:59:59Z"  
...   }  
... })  
960
```

En este caso, tanto los códigos como los resultados pueden verse en la misma captura.

Se ha aplicado un `.count()` filtrando por `location_id` para separar las dos ciudades, `sensor_id` para seleccionar de ambas ciudades el sensor con id 1 (que mide la temperatura) y un rango de días para contar los 10 días, incluido el décimo. Cabe destacar que el rango de fechas podría haber sido "2020-07-10T23:45:00Z" y el resultado habría sido el mismo, pero se decide mantener hasta el último segundo del día por convención.

Se podría haber incluido el filtro `"medidas.tipo_medida": "Temperatura"` para seleccionar los sensores correctos en lugar del `sensor_id`, y el resultado habría sido el mismo.

```
> db.datos_sensores.count({  
...   "location_id": 1,  
...   "medidas.tipo_medida": "Temperatura",  
...   "timestamp": {  
...     $gte: "2020-07-01T00:00:00Z",  
...     $lte: "2020-07-10T23:59:59Z"  
...   }  
... })  
960
```

Finalmente, se puede confirmar que falta un registro de este sensor en el caso de Sevilla, ya que su conteo da como resultado 959 en lugar de 960.

CASO 3: se pide identificar si hay algún documento que pueda distorsionar los resultados del estudio. Por ello, se quiere identificar posibles valores erróneos de temperatura enviados por el sensor.

Se pide, por tanto, en primer lugar, identificar los documentos de la colección “datos_sensores” que tengan una temperatura superior a 55 °C, tanto para Valladolid como para Sevilla (contar el número de casos en cada ciudad y mostrar también los documentos erróneos).

A la hora de mostrar los documentos con errores, solamente se devolverán las claves: *timestamp*, “location_id” y de la clave medidas mostrar solo el objeto correspondiente a la temperatura, sin mostrar el objeto de humedad relativa.

Una vez se hayan identificado, se procede a eliminar estos documentos de la colección, es decir: para ese *timestamp* solo quedará el documento correspondiente a la emisión de CO2 y el consumo eléctrico.

Además de las anteriores, se deberá adjuntar también una captura que demuestre que se han eliminado correctamente dichos documentos, si es que existiera alguno.

En primer lugar comprobamos si existen valores de temperatura superiores a 55°C registrados por los sensores de ambas ciudades.

```
> db.datos_sensores.count({  
...     "location_id": 2,  
...     "medidas": {  
...         $elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } }  
...     }  
...});  
1  
> db.datos_sensores.count({  
...     "location_id": 1,  
...     "medidas": {  
...         $elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } }  
...     }  
...});  
2
```

Se observa que existen 2 registros de temperatura con valor superior a 55°C en Valladolid y 1 en Sevilla. Se procede por tanto a mostrar la información solicitada acerca de estos registros, es decir, los elementos que coinciden con esos valores superiores a 55°C.

```
> db.datos_sensores.find({  
...     "location_id": 1,  
...     "medidas": {$elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } }}  
... },  
... {  
...     "timestamp": 1,  
...     "location_id": 1,  
...     "medidas": { $elemMatch: { "tipo_medida": "Temperatura" }}  
... }).pretty();
```

```

... }).pretty(),
{
    "_id" : ObjectId("66f70007b5db4a1c3f2493d2"),
    "timestamp" : "2020-07-05T17:15:00Z",
    "location_id" : 1,
    "medidas" : [
        {
            "tipo_medida" : "Temperatura",
            "valor" : 67.27,
            "unidad" : "°C"
        }
    ]
}
{
    "_id" : ObjectId("66f7004eb5db4a1c3f249cf4"),
    "timestamp" : "2020-07-11T19:30:00Z",
    "location_id" : 1,
    "medidas" : [
        {
            "tipo_medida" : "Temperatura",
            "valor" : 125.48,
            "unidad" : "°C"
        }
    ]
}

> db.datos_sensores.find({
...     "location_id": 2,
...     "medidas": { $elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } } }
... },
... {
...     "timestamp": 1,
...     "location_id": 1,
...     "medidas": { $elemMatch: { "tipo_medida": "Temperatura" } }
... }).pretty();
{
    "_id" : ObjectId("66f70030b5db4a1c3f24988d"),
    "timestamp" : "2020-07-08T03:00:00Z",
    "location_id" : 2,
    "medidas" : [
        {
            "tipo_medida" : "Temperatura",
            "valor" : 82.64,
            "unidad" : "°C"
        }
    ]
}

```

Se observa que, para Valladolid, existen dos registros con valores de temperatura de 67.27°C y 125.48°C, y para Sevilla, uno con valor de 82.64°C. Procedemos, por tanto, a eliminar la información relativa a la temperatura de estos documentos.

```

> db.datos_sensores.deleteMany({
...     "medidas": {
...         $elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } }
...     }
... });
{ "acknowledged" : true, "deletedCount" : 3 }

```

Con `.deleteMany()` se eliminan los documentos que contienen la condición establecida, en este caso, que en “medidas” tengan “tipo_medida”: “Temperatura” y “valor” mayor que 55. Para comprobar que estos documentos ya no existen, volvemos a realizar la consulta anterior que contaba el número de documentos con temperatura mayor que 55°C, pero esta vez sin filtrar por ciudad, lo que nos debería dar 3 en caso de no haber sido borrados, o 0 si los documentos fueron borrados con éxito.

```
> db.datos_sensores.count({  
...     "medidas": {  
...         $elemMatch: { "tipo_medida": "Temperatura", "valor": { $gt: 55 } }  
...     }  
...});  
0
```

Esta vez, no ha encontrado ningún documento con un valor mayor de 55 para la temperatura, comprobando ambas ciudades a la vez. Podemos realizar una última comprobación tratando de mostrar por pantalla alguno de los documentos borrados por id:

```
> db.datos_sensores.find({ "_id": ObjectId("66f70030b5db4a1c3f24988d") }).pretty();  
> |
```

Al filtrar por id y emplear una de las id de los documentos eliminados, mongo no da error, pero tampoco muestra ningún documento.

Dado que uno de los registros con temperatura errónea de Valladolid, así como el de Sevilla, tienen fecha anterior al día 11, estos registros estaban siendo contabilizados en el ejercicio anterior, por lo que, si volvemos a contar el número de documentos por ciudad para los diez primeros días, deberíamos obtener 959 en Valladolid y 958 en Sevilla, lo cuál nos sirve también para confirmar que han sido borrados.

```
> db.datos_sensores.count({  
...     "location_id": 1,  
...     "medidas.tipo_medida": "Temperatura",  
...     "timestamp": {  
...         $gte: "2020-07-01T00:00:00Z",  
...         $lte: "2020-07-10T23:59:59Z"  
...     }  
... })  
959  
> db.datos_sensores.count({  
...     "location_id": 2,  
...     "medidas.tipo_medida": "Temperatura",  
...     "timestamp": {  
...         $gte: "2020-07-01T00:00:00Z",  
...         $lte: "2020-07-10T23:59:59Z"  
...     }  
... })  
958
```

CASO 4: buscar el valor mínimo de temperatura en Sevilla. Se considera que es un valor poco realista para Sevilla y que es necesario multiplicarlo por un factor 1,2 para hacerlo un valor algo más real.

Nota: una vez hallado el ID del objeto a actualizar, investigar el funcionamiento de la función **\$mul** para comprobar si puede utilizarse en este caso para actualizar el valor o es necesario utilizar otro modificador.

Se deberá adjuntar una captura de la consulta de búsqueda junto con el documento al que le corresponde la temperatura mínima de Sevilla. Además, también se adjuntará la consulta de actualización y, por último, se mostrará todo el documento ya actualizado.

Lo primero será filtrar los documentos de Sevilla que tengan registros de temperatura. Descomponemos el array de medidas para tratar cada elemento como un documento separado y seleccionamos los documentos de temperatura. Luego ordenamos los valores de menor a mayor y limitamos la salida al primer elemento, la menor temperatura.

```
> db.datos_sensores.aggregate([
...   {$match: [
...     {"location_id": 2,
...      "medidas.tipo_medida": "Temperatura"}],
...   {$unwind: "$medidas" },
...   {$match: {"medidas.tipo_medida": "Temperatura"}},
...   {$sort: { "medidas.valor": 1 }},
...   {$limit: 1},
...   {$project: {"timestamp": 1, "location_id": 1, "medidas": 1}}
... ]).pretty();
{
  "_id" : ObjectId("66f70058b5db4a1c3f249e14"),
  "timestamp" : "2020-07-11T19:30:00Z",
  "location_id" : 2,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 9.14,
      "unidad" : "°C"
    }
]
}
```

Debido a que la función **\$mul** en MongoDB sólo permite multiplicar el valor de un campo numérico por otro número directamente en el documento y no sobre subdocumentos o arrays anidados dentro del documento, esta función no funcionará para modificar el campo del array de medidas. Por ello, se empleará el modificador **\$set** para establecer un valor actualizado en la temperatura para la id encontrada.

Lo primero es recoger en una variable el valor de la temperatura del documento encontrado (ya que no podemos actualizarla directamente con **\$mul** ni queremos tener que introducirla manualmente), y a continuación actualizamos el documento con la id encontrada empleando **\$set**. Finalmente, comprobamos el documento en cuestión y vemos que el dato referente a la temperatura se ha actualizado correctamente ($9.14 * 1.2 = 10.968$).

```

> var temp_min_sev = db.datos_sensores.findOne(
...   {
...     "_id": ObjectId("66f70058b5db4a1c3f249e14"),
...     "medidas.tipo_medida": "Temperatura"
...   },
...   {
...     "medidas": {
...       $elemMatch: { "tipo_medida": "Temperatura" }
...     }
...   }
... ).medidas[0].valor;
> temp_min_sev
9.14
>
> db.datos_sensores.updateOne(
...   {
...     "_id": ObjectId("66f70058b5db4a1c3f249e14"),
...     "medidas.tipo_medida": "Temperatura"
...   },
...   {
...     $set: { "medidas.$.valor": temp_min_sev * 1.2 }
...   }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.datos_sensores.find({ "_id": ObjectId("66f70058b5db4a1c3f249e14") }).pretty()
()
{
  "_id" : ObjectId("66f70058b5db4a1c3f249e14"),
  "timestamp" : "2020-07-11T19:30:00Z",
  "sensor_id" : 1,
  "location_id" : 2,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 10.968,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 33.42,
      "unidad" : "%"
    }
  ]
}

```

CASO 5: obtener los tres máximos valores de consumo eléctrico en un día de fin de semana en ambas ciudades y comparar sus valores (se considera fin de semana a partir de las 16:00 del viernes y hasta las 7:45 del lunes).

Devolver para cada ciudad un documento con el *timestamp* en que se producen esos máximos, el día de la semana y la hora a las que corresponden esos máximos y el valor del consumo eléctrico (subclave “tipo_medida” “Consumo_electrico” + subclave “valor” + subclave “unidad”, es decir: no se quieren obtener los valores de ese día de emisión de CO2).

En primer lugar se crean variables que recojan, con el formato adecuado, el inicio y el final de fin de semana a efectos de consumo eléctrico. A continuación, se filtran los datos por location_id, consumo eléctrico y el tiempo entre los límites establecidos para el fin de semana. Se ordenan los elementos por el valor de consumo eléctrico de manera descendente y se limita la salida a los 3 primeros (los 3 con mayor consumo). Finalmente, se organiza la salida tal como la queremos. Se procede de igual forma para el otro location_id y presentamos los resultados de las variables en las que hemos recogido el retorno de las consultas.

```
> var startWeekend = "2020-07-03T16:00:00Z";
> var endWeekend = "2020-07-06T07:45:00Z";
> var maxConsumoValladolid = db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 1,
...       "medidas.tipo_medida": "Consumo_electrico",
...       "timestamp": { $gte: startWeekend, $lte: endWeekend }
...     }
...   },
...   { $sort: { "medidas.valor": -1 } },
...   { $limit: 3 },
...   {
...     $project: {
...       "timestamp": 1,
...       "medidas": {
...         $filter: {
...           input: "$medidas",
...           as: "medida",
...           cond: { $eq: [ "$$medida.tipo_medida", "Consumo_electrico" ] }
...         }
...       }
...     }
...   }
... ]).pretty();
```

```
> var maxConsumoSevilla = db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 2,
...       "medidas.tipo_medida": "Consumo_electrico",
...       "timestamp": { $gte: startWeekend, $lte: endWeekend }
...     }
...   },
...   { $sort: { "medidas.valor": -1 } },
...   { $limit: 3 },
...   {
...     $project: {
...       "timestamp": 1,
...       "medidas": {
...         $filter: {
...           input: "$medidas",
...           as: "medida",
...           cond: { $eq: [ "$$medida.tipo_medida", "Consumo_electrico" ] }
...         }
...       }
...     }
...   }
... ]).pretty();
```

```
> maxConsumoValladolid
{
    "_id" : ObjectId("66f6ffffdb5db4a1c3f24925b"),
    "timestamp" : "2020-07-04T17:30:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00137,
            "unidad" : "kWh/m2"
        }
    ]
}
{
    "_id" : ObjectId("66f6ffffdb5db4a1c3f24926a"),
    "timestamp" : "2020-07-04T18:30:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00131,
            "unidad" : "kWh/m2"
        }
    ]
}
{
    "_id" : ObjectId("66f7000cb5db4a1c3f2493e9"),
    "timestamp" : "2020-07-05T18:30:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00136,
            "unidad" : "kWh/m2"
        }
    ]
}
```

```
> maxConsumoSevilla
{
    "_id" : ObjectId("66f70002b5db4a1c3f2492f9"),
    "timestamp" : "2020-07-04T09:45:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00192,
            "unidad" : "kWh/m2"
        }
    ]
}
{
    "_id" : ObjectId("66f70011b5db4a1c3f2494fb"),
    "timestamp" : "2020-07-05T17:45:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00187,
            "unidad" : "kWh/m2"
        }
    ]
}
{
    "_id" : ObjectId("66f70002b5db4a1c3f2492f5"),
    "timestamp" : "2020-07-04T09:30:00Z",
    "medidas" : [
        {
            "tipo_medida" : "Consumo_electrico",
            "valor" : 0.00192,
            "unidad" : "kWh/m2"
        }
    ]
}
```

CASO 6: se considera como perjudicial para la salud cualquier día en que el acumulado de las emisiones de CO₂ durante el día completo supere los 420 g CO₂/m². De los 14 días que comprende nuestro estudio, se pretende recuperar cuántos días se superó ese límite para Valladolid y, por otro lado, cuántos días se superó para Sevilla.

Se deben adjuntar tanto la captura del número de veces que se supera el límite para cada ciudad (clave “días_sobre_límite_permitido”), así como otra captura que muestre todos los documentos de Valladolid, por un lado, agrupados por la clave “dia_mes” y el acumulado total de emisión CO₂ bajo la clave “suma_Emisión_CO₂”, ordenados de mayor a menor emisión.

Presentar las mismas dos capturas para Sevilla y comentar los resultados. ¿Qué tienen los cuatro días de ambas ciudades en los que la emisión de CO₂ es más pequeña?

Las consultas serán idénticas para Valladolid y Sevilla, distinguiéndose únicamente por el filtrado sobre la clave “location_id”.

En primer lugar, buscamos el número de días que, tanto Valladolid como Sevilla, sobrepasaron los 420g CO₂/m² durante el periodo recogido por los sensores. Para Valladolid, filtramos por su id y por la medida que vamos a estudiar, descomponemos el array de medidas para asegurarnos de seleccionar sólo los documentos con registros sobre emisiones de CO₂ y los seleccionamos, agrupamos por día y sumamos el valor de las emisiones, filtramos para seleccionar sólo las que superan los 420g CO₂/m² y finalmente hacemos un recuento.

Repetimos lo mismo para la location_id de Sevilla.

```
> db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 1,
...       "medidas.tipo_medida": "Emisión_CO2"
...     }
...   },
...   { $unwind: "$medidas" },
...   { $match: {
...     "medidas.tipo_medida": "Emisión_CO2"
...   }},
...   { $group: {
...     _id: { day: { $substr: ["$timestamp", 0, 10] } },
...     totalCO2: { $sum: "$medidas.valor" }
...   }}
... ],
... { $match: { totalCO2: { $gt: 420 } }
... },
... { $count: "días_sobre_límite_permitido"
... }
... ]);
{ "días_sobre_límite_permitido" : 4 }
```

```

> db.datos_sensores.aggregate([
...   {
...     $match: {
...       "location_id": 2,
...       "medidas.tipo_medida": "Emision_CO2"
...     }
...   },
...   { $unwind: "$medidas" },
...   { $match: {
...     "medidas.tipo_medida": "Emision_CO2"
...   }},
...   { $group: {
...     _id: { day: { $substr: ["$timestamp", 0, 10] } },
...     totalCO2: { $sum: "$medidas.valor" }
...   }}
... ],
... { $match: { totalCO2: { $gt: 420 } } },
... [
...   { $count: "dias_sobre_limite_permitido"
... }
... ],
... [
...   { "dias_sobre_limite_permitido" : 8 }
... ]);
{ "dias_sobre_limite_permitido" : 8 }

```

Se puede comprobar que, para Valladolid, existen 4 días en que superaron el límite a partir del cuál se considera perjudicial para la salud en el periodo estudiado. Para Sevilla fueron 8 días.

A continuación, se agrupan todos los documentos de Valladolid por un lado y de Sevilla por otro y se muestran las acumulaciones de CO2 diarias. Se filtran por location_id y emisiones de CO2, se añade un campo de fecha para formatear posteriormente la clave dia_mes, filtramos las medidas que queremos seleccionar, agrupamos por dia_mes y sumamos los valores de las medidas correspondientes a las emisiones de CO2. A continuación, ordenamos de manera descendente, y presentamos los resultados de la consulta.

Repetimos el proceso para obtener las acumulaciones de emisiones de CO2 diarias para Sevilla.

```

> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 1,
...     "medidas.tipo_medida": "Emision_CO2"}},
...   { $addFields: {
...     "timestamp_date": {
...       $dateFromString: {
...         dateString: "$timestamp"}}}},
...   { $project: {
...     "dia_mes": { $dateToString: { format: "%d-%m", date: "$timestamp_date" } },
...     "medidas": {
...       $filter: { input: "$medidas", as: "medida", cond: { $eq: [ $$medida.tipo_medida, "Emision_CO2"] }}}}},
...   { $group: {
...     _id: "$dia_mes",
...     suma_Emission_CO2: { $sum: { $arrayElemAt: ["$medidas.valor", 0] }}},
...   { $sort: { suma_Emission_CO2: -1 }},
...   { $project: {
...     dia_mes: "$_id",
...     id: 0,
...     suma_Emission_CO2: 1}}]);
{ "suma_Emission_CO2": 425.432, "dia_mes": "09-07" }

```

```

> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 2,
...     "medidas.tipo_medida": "Emision_CO2"}},
...   { $addFields: {
...     "timestamp_date": {
...       $dateFromString: {
...         dateString: "$timestamp"}}}},
...   { $project: {
...     "dia_mes": { $dateToString: { format: "%d-%m", date: "$timestamp_date" } },
...     "medidas": {
...       $filter: { input: "$medidas", as: "medida", cond: { $eq: ["$medida.tipo_medida", "Emision_CO2"] }}}}},
...   { $group: {
...     id: "$dia_mes",
...     suma_Emission_CO2: { $sum: { $arrayElemAt: ["$medidas.valor", 0] }}},
...   { $sort: { suma_Emission_CO2: -1 }},
...   { $project: {
...     dia_mes: "$_id",
...     id: 0,
...     suma_Emission_CO2: 1}}]);

```

...

suma_Emission_CO2	dia_mes
425.432	09-07
422.799	13-07
421.447	07-07
420.337	02-07
418.368	06-07
417.688	08-07
416.577	14-07
411.776	01-07
374.376	10-07
369.989	03-07
164.694	11-07
164.522	05-07
164.474	04-07
163.997	12-07

suma_Emission_CO2	dia_mes
509.796	01-07
507.367	07-07
503.715	06-07
501.444	02-07
501.261	09-07
500.754	13-07
499.94100000000003	08-07
483.623	14-07
364.4	03-07
362.891	10-07
203.911	12-07
203.811	04-07
203.318	11-07
203.205	05-07

Como conclusión general, se puede afirmar que, al menos en el periodo estudiado, Sevilla emite bastante más CO2 diariamente que Valladolid.

Respondiendo a la pregunta, para ambas ciudades, los 4 registros de emisiones menores tienen dos lecturas: una, que existe un descenso considerable entre estos 4 registros y el primero del resto (es decir, el quinto menor acumulado) para ambas ciudades, y otra que los días son los

mismos en ambas ciudades: 04, 05, 11 y 12. Teniendo en cuenta que son días correlativos de dos en dos, y que entre ambos conjuntos existen 7 días de diferencia, es fácil adivinar que se trata de los fines de semana (sábados 04 y 11 y domingos 05 y 12 de julio). En ambos casos, el descenso en las emisiones de CO₂ en fin de semana se debe al descanso de los trabajadores y a la no actividad.

CASO 7: obtener la media de emisiones de CO2 de cada ciudad por separado en la hora punta de personas en la oficina, que se considera las 10 AM. Por tanto, se obtendrán por un lado los registros con hora 10 en el *timestamp*, que midan CO2 y que sean de Valladolid y obtener la media de emisiones de CO2 durante los 14 días en esa hora como la clave “Avg_Emission_CO2” (recordar que cada hora se reciben cuatro valores desde el sensor para cada ciudad). Mostrar los resultados ordenados por la clave “Avg_Emission_CO2”, mostrando como primer documento la media más baja. Redondear esta clave a dos decimales. Realizar la misma operativa con Sevilla.

En primer lugar filtramos por location_id para obtener los datos de Valladolid, el tipo de medida que vamos a emplear y las fechas que tengan hora 10. Descomponemos el array y filtramos los documentos de emisión de CO2. A continuación, agrupamos por fecha (eliminando al mismo tiempo la referencia horaria) y calculamos la media de los valores de CO2 para estos documentos. Antes de finalizar, ordenamos de manera ascendente para que el primer documento mostrado tenga la media más baja.

Para Valladolid:

```
> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 1,
...     "medidas.tipo_medida": "Emision_CO2",
...     $expr: { $eq: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 10] }},
...   { $unwind: "$medidas" },
...   { $match: {
...     "medidas.tipo_medida": "Emision_CO2"}},
...   { $group: {
...     _id: { $substr: ["$timestamp", 0, 10] },
...     Avg_Emission_CO2: { $avg: "$medidas.valor" }}},
...   { $project: {
...     _id: 0,
...     Fecha: "$_id",
...     Avg_Emission_CO2: { $round: ["$Avg_Emission_CO2", 2] }}},
...   { $sort: { Avg_Emission_CO2: 1 }}]);
{
  "fecha" : "2020-07-11", "Avg_Emission_CO2" : 1.75 }
{
  "fecha" : "2020-07-12", "Avg_Emission_CO2" : 1.79 }
{
  "fecha" : "2020-07-04", "Avg_Emission_CO2" : 1.81 }
{
  "fecha" : "2020-07-05", "Avg_Emission_CO2" : 1.83 }
{
  "fecha" : "2020-07-07", "Avg_Emission_CO2" : 8.46 }
{
  "fecha" : "2020-07-01", "Avg_Emission_CO2" : 8.81 }
{
  "fecha" : "2020-07-06", "Avg_Emission_CO2" : 8.93 }
{
  "fecha" : "2020-07-02", "Avg_Emission_CO2" : 9.07 }
{
  "fecha" : "2020-07-10", "Avg_Emission_CO2" : 9.12 }
{
  "fecha" : "2020-07-14", "Avg_Emission_CO2" : 9.15 }
{
  "fecha" : "2020-07-03", "Avg_Emission_CO2" : 9.17 }
{
  "fecha" : "2020-07-09", "Avg_Emission_CO2" : 9.24 }
{
  "fecha" : "2020-07-08", "Avg_Emission_CO2" : 9.31 }
{
  "fecha" : "2020-07-13", "Avg_Emission_CO2" : 9.51 }
```

Una alternativa sería emplear la siguiente expresión para forzar a que se asegure de que no sólo se seleccionan los registros con fecha-hora de las 10 en punto, sino todas los registros producidos entre las 10:00 y las 10:59, pero el resultado es el mismo, por lo que damos por supuesto que la expresión utilizada ya está seleccionando bien los documentos:

```
...
  $expr: {
    $and: [
      { $eq: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 10] },
      { $lte: [{ $minute: { $dateFromString: { dateString: "$timestamp" } } }, 59]}]}},
```

Para Sevilla:

```
> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 2,
...     "medidas.tipo_medida": "Emision_CO2",
...     $expr: { $eq: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 10] }}},
...   { $unwind: "$medidas" },
...   { $match: {
...     "medidas.tipo_medida": "Emision_CO2"}},
...   { $group: {
...     _id: { $substr: ["$timestamp", 0, 10] },
...     Avg_Emission_CO2: { $avg: "$medidas.valor" }}},
...   { $project: {
...     _id: 0,
...     fecha: "$_id",
...     Avg_Emission_CO2: { $round: ["$Avg_Emission_CO2", 2] }}},
...   { $sort: { Avg_Emission_CO2: 1 }}];
{
  "fecha" : "2020-07-11", "Avg_Emission_CO2" : 2.2
}
{
  "fecha" : "2020-07-05", "Avg_Emission_CO2" : 2.22
}
{
  "fecha" : "2020-07-12", "Avg_Emission_CO2" : 2.22
}
{
  "fecha" : "2020-07-04", "Avg_Emission_CO2" : 2.24
}
{
  "fecha" : "2020-07-10", "Avg_Emission_CO2" : 8
}
{
  "fecha" : "2020-07-03", "Avg_Emission_CO2" : 8.12
}
{
  "fecha" : "2020-07-08", "Avg_Emission_CO2" : 9.67
}
{
  "fecha" : "2020-07-07", "Avg_Emission_CO2" : 9.74
}
{
  "fecha" : "2020-07-14", "Avg_Emission_CO2" : 10.04
}
{
  "fecha" : "2020-07-06", "Avg_Emission_CO2" : 10.17
}
{
  "fecha" : "2020-07-02", "Avg_Emission_CO2" : 10.23
}
{
  "fecha" : "2020-07-13", "Avg_Emission_CO2" : 10.42
}
{
  "fecha" : "2020-07-09", "Avg_Emission_CO2" : 10.51
}
{
  "fecha" : "2020-07-01", "Avg_Emission_CO2" : 10.62
}
```

CASO 8: Se ha descubierto que el sensor de temperatura de Valladolid mide 1,5 °C de más. Por ello, se pide actualizar todos los valores correspondientes a este sensor decrementando el valor de la temperatura en 1,5 °C.

Antes de realizar la actualización, se ordenarán mostrando primero el de mayor temperatura, y muestra los dos primeros documentos del sensor de Valladolid con la mayor temperatura. Solo mostrar las siguientes claves: timestamp, "location_id". Del array de medidas solo mostrar el primer ítem, por ejemplo: "medidas" : [{ "tipo_medida" : "Temperatura", "valor" : 15.75, "unidad" : "°C" }]. No incluir el ObjectId).

Realizar el mismo proceso una vez que has actualizado los valores y realiza capturas de los dos documentos anteriores para comprobar que se han actualizado.

Para seleccionar los dos documentos con la temperatura más alta de Valladolid tenemos que filtrar por ambas claves-valor, luego descomponemos el array de medidas y seleccionamos el elemento correspondiente a la temperatura, ordenamos de manera descendente y limitamos la salida a dos elementos. Finalmente, presentamos la información que nos interesa.

```
> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 1,
...     "medidas.tipo_medida": "Temperatura"}},
...   { $unwind: "$medidas" },
...   { $match: {"medidas.tipo_medida": "Temperatura"}},
...   { $sort: {"medidas.valor": -1}},
...   { $limit: 2},
...   { $project: {
...     id: 0,
...     timestamp: 1,
...     location_id: 1,
...     medidas: {
...       tipo_medida: 1,
...       valor: 1,
...       unidad: 1}}}}].pretty()
```

```
{ "timestamp" : "2020-07-14T17:30:00Z",
  "location_id" : 1,
  "medidas" : {
    "tipo_medida" : "Temperatura",
    "valor" : 40.88,
    "unidad" : "°C"
  }
}
{
  "timestamp" : "2020-07-03T16:30:00Z",
  "location_id" : 1,
  "medidas" : {
    "tipo_medida" : "Temperatura",
    "valor" : 40.87,
    "unidad" : "°C"
  }
}
```

Podemos comprobar que, para Valladolid, las dos temperaturas más altas se corresponden a las de los días 14 de julio a las 17:30 y 03 de julio a las 16:30, siendo 40.88°C y 40.87°C respectivamente.

A continuación, aplicamos un decremento de 1.5°C a todos los registros de temperatura de Valladolid. Filtramos los documentos que vamos a modificar, empleamos \$inc para aplicar un incremento (será negativo, es decir, un decremento) y filtramos del array el elemento con tipo_medida "Temperatura". Vemos según la salida de la consola que se han actualizado 1342 elementos.

```
> db.datos_sensores.updateMany(  
...   {  
...     "location_id": 1,  
...     "sensor_id": 1,  
...     "medidas": {  
...       $elemMatch: { "tipo_medida": "Temperatura" }  
...     },  
...     { $inc: { "medidas.$[elem].valor": -1.5 }},  
...     { arrayFilters: [{ "elem.tipo_medida": "Temperatura" }]}  
{ "acknowledged" : true, "matchedCount" : 1342, "modifiedCount" : 1342 }
```

Volvemos a emplear el mismo código del inicio del caso para comprobar que los valores se han actualizado correctamente y vemos que los registros con mayor temperatura de Valladolid se siguen correspondiendo con las mismas fechas, pero ahora sus valores son 39.38°C y 39.37°C respectivamente, 1.5 menores que al inicio. Se acepta, por tanto, que las modificaciones se han aplicado correctamente.

```
{  
  "timestamp" : "2020-07-14T17:30:00Z",  
  "location_id" : 1,  
  "medidas" : {  
    "tipo_medida" : "Temperatura",  
    "valor" : 39.38,  
    "unidad" : "°C"  
  }  
}  
{  
  "timestamp" : "2020-07-03T16:30:00Z",  
  "location_id" : 1,  
  "medidas" : {  
    "tipo_medida" : "Temperatura",  
    "valor" : 39.37,  
    "unidad" : "°C"  
  }  
}
```

CASO 9: se quieren analizar los porcentajes de humedad relativa en horario laboral (8-18:00) de ambas ciudades. Recuperar por un lado los cinco documentos con los valores mínimos de humedad relativa en Sevilla y los cinco documentos con humedad relativa mínima en Valladolid por separado (ambos en horario laboral).

Sólo se quieren recuperar de estos documentos el *timestamp* y la subclave **medidas.tipo_medida = Humedad_relativa**, junto con el valor y la unidad asociados.

Analizar si los mínimos se producen siempre en la misma franja horaria (mismo intervalo de 1-2 horas, o es variable). Comentar si varían de una ciudad a otra.

Para obtener los porcentajes de humedad relativa de Valladolid filtramos por su location_id y el tipo de medida que buscamos, añadimos una expresión que limite las fechas a aquellas que tengan la hora entre 08 y 18, descomponemos el array de medidas y seleccionamos el elemento correspondiente a la humedad relativa, ordenamos ascendente, es decir, el menor primero, y limitamos la salida a los 5 primeros documentos.

Para Sevilla se procede de la misma manera, cambiando location_id en el filtro inicial.

```
> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 1,
...     "medidas.tipo_medida": "Humedad_relativa",
...     $expr: {
...       $and: [
...         { $gte: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 8] },
...         { $lte: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 18] }]}},
...   { $unwind: "$medidas" },
...   { $match: {"medidas.tipo_medida": "Humedad_relativa" } },
...   { $sort: { "medidas.valor": 1 } },
...   { $limit: 5 },
...   { $project: {
...     _id: 0,
...     timestamp: 1,
...     "medidas.tipo_medida": 1,
...     "medidas.valor": 1,
...     "medidas.unidad": 1}
...   ]}.pretty()
{
```

```
{
  "timestamp" : "2020-07-03T16:45:00Z",
  "medidas" : {
    "tipo_medida" : "Humedad_relativa",
    "valor" : 5.74,
    "unidad" : "%"
  }
}
{
  "timestamp" : "2020-07-06T17:00:00Z",
  "medidas" : {
    "tipo_medida" : "Humedad_relativa",
    "valor" : 6.81,
    "unidad" : "%"
  }
}
{
  "timestamp" : "2020-07-10T17:45:00Z",
  "medidas" : {
    "tipo_medida" : "Humedad_relativa",
    "valor" : 7.6,
    "unidad" : "%"
  }
}
```

```
{
    "timestamp" : "2020-07-14T17:30:00Z",
    "medidas" : {
        "tipo_medida" : "Humedad_relativa",
        "valor" : 8.14,
        "unidad" : "%"
    }
}
{
    "timestamp" : "2020-07-07T14:45:00Z",
    "medidas" : {
        "tipo_medida" : "Humedad_relativa",
        "valor" : 8.53,
        "unidad" : "%"
    }
}
```

```
> db.datos_sensores.aggregate([
...   { $match: {
...     "location_id": 2,
...     "medidas.tipo_medida": "Humedad_relativa",
...     $expr: {
...       $and: [
...         { $gte: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 8] },
...         { $lte: [{ $hour: { $dateFromString: { dateString: "$timestamp" } } }, 18] }]}},
...     { $unwind: "$medidas" },
...     { $match: {"medidas.tipo_medida": "Humedad_relativa"  }},
...     { $sort: { "medidas.valor": 1 }},
...     { $limit: 5 },
...     { $project: {
...       _id: 0,
...       timestamp: 1,
...       "medidas.tipo_medida": 1,
...       "medidas.valor": 1,
...       "medidas.unidad": 1}}
...   ]}).pretty()
{
    "timestamp" : "2020-07-07T14:45:00Z"
}
```

```
{
    "timestamp" : "2020-07-05T14:00:00Z",
    "medidas" : {
        "tipo_medida" : "Humedad_relativa",
        "valor" : 0.3,
        "unidad" : "%"
    }
}
{
    "timestamp" : "2020-07-01T16:15:00Z",
    "medidas" : {
        "tipo_medida" : "Humedad_relativa",
        "valor" : 0.61,
        "unidad" : "%"
    }
}
{
    "timestamp" : "2020-07-02T15:15:00Z",
    "medidas" : {
        "tipo_medida" : "Humedad_relativa",
        "valor" : 0.94,
        "unidad" : "%"
    }
}
```

```
{  
    "timestamp" : "2020-07-14T14:45:00Z",  
    "medidas" : {  
        "tipo_medida" : "Humedad_relativa",  
        "valor" : 1.25,  
        "unidad" : "%"  
    }  
}  
{  
    "timestamp" : "2020-07-10T13:45:00Z",  
    "medidas" : {  
        "tipo_medida" : "Humedad_relativa",  
        "valor" : 2.1,  
        "unidad" : "%"  
    }  
}
```

En el caso de Valladolid, las humedades relativas mínimas van de 5.74% a 8.53% y se producen entre las 14:45 y las 17:45 (cuatro de ellas se producen en un rango de una hora, entre las 16:45 y las 17:45), mientras en Sevilla van de 0.3% a 2.1% y se producen entre las 13:45 y las 16:15.

Se puede concluir que las amplitudes de estos rangos horarios son similares (amplitud de 3 horas en el caso de Valladolid y de 2.5 horas en el de Sevilla), siendo también unas franjas horarias similares, aunque algo más tardías en el caso de Valladolid (una hora más tardía en su límite inferior y 1.5 horas más tardía en su límite superior).

CASO 10: se quieren actualizar todos los documentos para introducir en el `array` de medidas dos nuevos elementos que van a ser constantes y que van a tener el valor del precio del kWh y de la superficie total de la sede.

Solamente se añadirán a los documentos que tengan como medida el consumo eléctrico (los que tienen temperatura y humedad no).

• Los ítems a introducir son para Valladolid:

```
{"precio_kWh":0.102,"unidad":"€/kWh"}, {"superficie":450,"unidad":"m2"}
```

• Los ítems a introducir son para Sevilla:

```
{"precio_kWh":0.107,"unidad":"€/kWh"}, {"superficie":550,"unidad":"m2"}
```

Para comprobar que se ha actualizado correctamente la colección, muestra cuatro documentos de Sevilla y cuatro de Valladolid, correspondientes a los *timestamps*: **2020-07-01T08:00:00Z**, **2020-07-01T23:15:00Z**. Cada intervalo se envían dos documentos desde cada ciudad: (**Temperatura y Humedad_relativa**) + (**Emision_CO2 y Consumo_electrico**).

Para realizar la modificación del `array`, primero filtramos por `location_id` de la ciudad y el tipo_medida que necesitamos encontrar, introducimos los cambios mediante `$push` (necesitamos emplear un `$each` porque vamos a introducir más de un elemento distinto en el `array`) y creamos los datos a introducir. Se produce la actualización de 1344 elementos en ambos casos.

```
> db.datos_sensores.updateMany(  
...   {  
...     "location_id": 1,  
...     "medidas.tipo_medida": "Consumo_electrico"  
...   },  
...   {  
...     $push: {  
...       "medidas": {  
...         $each: [  
...           { "tipo_medida": "precio_kwh", "valor": 0.102, "unidad": "€/kWh" },  
...           { "tipo_medida": "superficie", "valor": 450, "unidad": "m2" }]]}  
...   });  
{ "acknowledged" : true, "matchedCount" : 1344, "modifiedCount" : 1344 }
```

```
> db.datos_sensores.updateMany(  
...   {  
...     "location_id": 2,  
...     "medidas.tipo_medida": "Consumo_electrico"  
...   },  
...   {  
...     $push: {  
...       "medidas": {  
...         $each: [  
...           { "tipo_medida": "precio_kWh", "valor": 0.107, "unidad": "€/kWh" },  
...           { "tipo_medida": "superficie", "valor": 550, "unidad": "m2" }]]}  
...   });  
{ "acknowledged" : true, "matchedCount" : 1344, "modifiedCount" : 1344 }
```

A continuación, comprobamos 4 documentos de cada ciudad seleccionando 2 fechas concretas (los cuáles contienen 2 documentos cada una: uno referente a la temperatura y la humedad relativa, que no debe haber sufrido cambios, y otro referente a la emisión de CO2 y al consumo eléctrico, donde ahora deben aparecer los 2 nuevos elementos introducidos).

Comprobaciones para Valladolid:

```
> db.datos_sensores.find(
...   {
...     "location_id": 1,
...     "timestamp": { $in: ["2020-07-01T08:00:00Z", "2020-07-01T23:15:00Z"] }
...   },
...   {
...     _id: 0,
...     timestamp: 1,
...     location_id: 1,
...     medidas: 1
...   }
... ).limit(4).pretty();
{
  "timestamp" : "2020-07-01T08:00:00Z",
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 18.7,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 77.36,
      "unidad" : "%"
    }
  ]
}
```

```
{
  "timestamp" : "2020-07-01T08:00:00Z",
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Emision_CO2",
      "valor" : 6.7,
      "unidad" : "gCO2/m2"
    },
    {
      "tipo_medida" : "Consumo_electrico",
      "valor" : 0.01788,
      "unidad" : "kwh/m2"
    },
    {
      "tipo_medida" : "precio_kwh",
      "valor" : 0.102,
      "unidad" : "€/kWh"
    },
    {
      "tipo_medida" : "superficie",
      "valor" : 450,
      "unidad" : "m2"
    }
  ]
}
```

```
{
  "timestamp" : "2020-07-01T23:15:00Z",
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 10.06,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 85.96,
      "unidad" : "%"
    }
  ]
}
```

```
{
  "timestamp" : "2020-07-01T23:15:00Z",
  "location_id" : 1,
  "medidas" : [
    {
      "tipo_medida" : "Emision_CO2",
      "valor" : 1.629,
      "unidad" : "gCO2/m2"
    },
    {
      "tipo_medida" : "Consumo_electrico",
      "valor" : 0.00164,
      "unidad" : "kWh/m2"
    },
    {
      "tipo_medida" : "precio_kWh",
      "valor" : 0.102,
      "unidad" : "€/kWh"
    },
    {
      "tipo_medida" : "superficie",
      "valor" : 450,
      "unidad" : "m2"
    }
  ]
}
```

Se constata que, para Valladolid, los documentos cuyas medidas hacen referencia a la emisión de CO2 y al consumo eléctrico, ahora también tienen información sobre el precio del kWh y la superficie del local de la empresa, mientras los que hacen referencia a la temperatura y la humedad relativa permanecen inalterados.

Comprobaciones para Sevilla:

```
> db.datos_sensores.find(
...   {
...     "location_id": 2,
...     "timestamp": { $in: ["2020-07-01T08:00:00Z", "2020-07-01T23:15:00Z"] }
...   },
...   {
...     _id: 0,
...     timestamp: 1,
...     location_id: 1,
...     medidas: 1
...   }
... ).limit(4).pretty();
{
  "timestamp" : "2020-07-01T08:00:00Z",
  "location_id" : 2,
  "medidas" : [
    {
      "tipo_medida" : "Temperatura",
      "valor" : 22.6,
      "unidad" : "°C"
    },
    {
      "tipo_medida" : "Humedad_relativa",
      "valor" : 40.73,
      "unidad" : "%"
    }
  ]
}
```

```
{  
    "timestamp" : "2020-07-01T08:00:00Z",  
    "location_id" : 2,  
    "medidas" : [  
        {  
            "tipo_medida" : "Emision_CO2",  
            "valor" : 10.262,  
            "unidad" : "gCO2/m2"  
        },  
        {  
            "tipo_medida" : "Consumo_electrico",  
            "valor" : 0.01467,  
            "unidad" : "kWh/m2"  
        },  
        {  
            "tipo_medida" : "precio_kWh",  
            "valor" : 0.107,  
            "unidad" : "€/kWh"  
        },  
        {  
            "tipo_medida" : "superficie",  
            "valor" : 550,  
            "unidad" : "m2"  
        }  
    ]  
}
```

```
{  
    "timestamp" : "2020-07-01T23:15:00Z",  
    "location_id" : 2,  
    "medidas" : [  
        {  
            "tipo_medida" : "Temperatura",  
            "valor" : 25.58,  
            "unidad" : "°C"  
        },  
        {  
            "tipo_medida" : "Humedad_relativa",  
            "valor" : 30.86,  
            "unidad" : "%"   
        }  
    ]  
}
```

```
{  
    "timestamp" : "2020-07-01T23:15:00Z",  
    "location_id" : 2,  
    "medidas" : [  
        {  
            "tipo_medida" : "Emision_CO2",  
            "valor" : 1.992,  
            "unidad" : "gCO2/m2"  
        },  
        {  
            "tipo_medida" : "Consumo_electrico",  
            "valor" : 0.00259,  
            "unidad" : "kWh/m2"  
        },  
        {  
            "tipo_medida" : "precio_kWh",  
            "valor" : 0.107,  
            "unidad" : "€/kWh"  
        },  
        {  
            "tipo_medida" : "superficie",  
            "valor" : 550,  
            "unidad" : "m2"  
        }  
    ]  
}
```

Al igual que en el caso de Valladolid, los cambios se han introducido con éxito.