MongoDB – HOWTO-5B

Este documento constituye una ampliación del documento "**MongoDB – HOWTO-5**". Incluye ejemplos de uso y, en algunos casos, sentencias quivalentes en versión SQL.

En los ejemplos que aparecen en este documento se ha utilizado una colección de datos con información de series de población de los municipios españoles entre 2000 y 2017 obtenida del INE. Esta información se encuentra en el fichero **poblacion.csv** y puede ser cargada en la colección **poblacion** de la base de datos **test** de nuestro servidor de datos MongoDB mediante el siguiente comando:

```
mongoimport --type csv --db test --collection poblacion --headerline --file poblacion.csv
```

Las sentencias de algunos de los ejemplos incluyen al final la función '**pretty()**', para poder visualizar mejor los datos.

$project (SELECT)

Selecciona los campos de los documentos en la etapa de la secuencia de agregación en que se encuentre, agregando nuevos campos o eliminando campos existentes, y los pasa a la siguiente etapa de la secuencia de agregación.

Sintaxis:

```
{
    $project: {
        <specification(s)>
    }
}
```

Ejemplo:

Mostrar el nombre de provincia, nombre de municipio y población total en 2017 de todos los municipios de España.

Además, renombrar los nombres de los datos 'NOM_PROVINCIA', 'NOM_MUNICIPIO' y '2017T' por 'Provincia', 'Municipio' y 'Población' respectivamente.

```
db.poblacion.aggregate(
    [
        {
            $project: {
                Provincia:'$NOM_PROVINCIA',
                Municipio:'$NOM_MUNICIPIO',
                Población:'$2017T',
                _id:0
            }
        }
    ]
)
```

Explicación:
- Selecciona los campos: 'NOM_PROVINCIA', 'NOM_MUNICIPIO', '2017T', renombrándolos a 'Provincia', 'Municipio' y 'Población' respectivamente. Oculta '_id'.

Resultado:


```
{ "Provincia" : "Salamanca", "Municipio" : "Linares de Riofrío", "Población" : 979 }
{ "Provincia" : "Salamanca", "Municipio" : "Lumbrales", "Población" : 1662 }
{ "Provincia" : "Salamanca", "Municipio" : "Macotera", "Población" : 1114 }
{ "Provincia" : "Salamanca", "Municipio" : "Machacón", "Población" : 449 }
{ "Provincia" : "Salamanca", "Municipio" : "Madroñal", "Población" : 146 }
{ "Provincia" : "Salamanca", "Municipio" : "Maíllo (El)", "Población" : 279 }
{ "Provincia" : "Salamanca", "Municipio" : "Malpartida", "Población" : 101 }
{ "Provincia" : "Salamanca", "Municipio" : "Mancera de Abajo", "Población" : 231 }
{ "Provincia" : "Salamanca", "Municipio" : "Manzano (El)", "Población" : 75 }
{ "Provincia" : "Salamanca", "Municipio" : "Martiago", "Población" : 292 }
...
```


SQL:

```
SELECT NOM_PROVINCIA as Provincia,
       NOM_MUNICIPIO as Municipio,
       '2017T' as 'Población'
  FROM poblacion;
```

$match (WHERE, HAVING)

Filtra los documentos que pasan a la siguiente etapa de la secuencia de agregación, permitiendo que sólo lo hagan los que coincidan con las condiciones del filtro.

Sintaxis:

```
{
    $match: {
        <query>
    }
}
```

Ejemplo:

Mostrar el nombre de provincia, nombre de municipio y población total en 2017 de aquellos municipios de España que en el año 2017 tenían 500.000 habitantes ó más.

Además, renombrar los nombres de los datos 'NOM_PROVINCIA', 'NOM_MUNICIPIO' y '2017T' por 'Provincia', 'Municipio' y 'Población' respectivamente.

```
db.poblacion.aggregate(
    [
        {
            $match : {
                '2017T': { $gte: 500000 }
            }
        },
        {
```

```
            $project: {
                Provincia:'$NOM_PROVINCIA',
                Municipio:'$NOM_MUNICIPIO',
                Población:'$2017T',
                _id:0
            }
        }
    ]
).pretty()
```

Explicacicón:
- Selecciona los documentos con población total >= 500.000 habitantes en 2017.
- Selecciona los campos: 'NOM_PROVINCIA', 'NOM_MUNICIPIO', '2017T', renombrándolos a 'Provincia', 'Municipio' y 'Población' respectivamente. Oculta '_id'.

Resultado:

```
{ "Provincia" : "Barcelona", Municipio" : "Barcelona", "Población" : 1620809 }
{ "Provincia" : "Valencia/València", "Municipio" : "Valencia", "Población" : 787808 }
{ "Provincia" : "Málaga", "Municipio" : "Málaga", "Población" : 569002 }
{ "Provincia" : "Madrid", "Municipio" : "Madrid", "Población" : 3182981 }
{ "Provincia" : "Zaragoza", "Municipio" : "Zaragoza", "Población" : 664938 }
{ "Provincia" : "Sevilla", "Municipio" : "Sevilla", "Población" : 689434 }
```

SQL:

```
SELECT NOM_PROVINCIA as Provincia,
       NOM_MUNICIPIO as Municipio,
       '2017T' as 'Población'
  FROM poblacion
 WHERE '2017T' > = 500000;
```

$sort (ORDER BY)

Reordena mediante una clave especificada los documentos que pasan a la siguiente etapa de la secuencia de agregación.

Los valores de \<sort ord> pueden ser:

Valor	Descripción
1	sentido ascencente: de menor a mayor
-1	sentido decencente: de mayor a menor

Sintaxis:

```
{
    $sort: {
        <field1>: <sort order>,
        <field2>: <sort order>
        ...
    }
```

```
}
```

Ejemplo:

Mostrar el nombre de provincia, nombre de municipio y población total en 2017 de aquellos municipios de España que en el año 2017 tenían 500.000 habitantes ó más, ordenándolos por número de habitantes en sentido descendente.

Además, renombrar los nombres de los datos 'NOM_PROVINCIA', 'NOM_MUNICIPIO' y '2017T' por 'Provincia', 'Municipio' y 'Población' respectivamente.

```
db.poblacion.aggregate(
    [
        {
            $match : { '2017T': { $gte: 500000 } }
        },
        {
            $project: {
                Provincia:'$NOM_PROVINCIA',
                Municipio:'$NOM_MUNICIPIO',
                Población:'$2017T',
                _id:0
            }
        },
        {
            $sort: {
                Población:-1
            }
        }
    ]
).pretty()
```

Explicación:
- Selecciona los documentos con población total >= 500.000 habitantes en 2017.
- Selecciona los campos: 'NOM_PROVINCIA', 'NOM_MUNICIPIO', '2017T', renombrándolos a 'Provincia', 'Municipio' y 'Población' respectivamente. Oculta '_id'.
- Ordena los documentos por total de población en sentido descendente.

Resultado:

```
{ "Provincia" : "Madrid", "Municipio" : "Madrid", "Población" : 3182981 }
{ "Provincia" : "Barcelona", "Municipio" : "Barcelona", "Población" : 1620809 }
{ Provincia" : "Valencia/València", "Municipio" : "Valencia", "Población" : 787808 }
{ "Provincia" : "Sevilla", "Municipio" : "Sevilla", "Población" : 689434 }
{ "Provincia" : "Zaragoza", "Municipio" : "Zaragoza", "Población" : 664938 }
{ "Provincia" : "Málaga", "Municipio" : "Málaga", "Población" : 569002 }
```

SQL:

```
  SELECT NOM_PROVINCIA as Provincia,
         NOM_MUNICIPIO as Municipio,
         '2017T' as 'Población'
    FROM poblacion
   WHERE '2017T' > = 500000
ORDER BY '2017T' desc;
```

$count (COUNT)

Devuelve el número de documentos en la etapa de la secuencia de agregación donde se encuentre.

Sintaxis:

```
{
    $count: <string>
}
```

Ejemplo:

Obtener el número total de municipios de España que en el año 2017 tenían 300.000 habitantes ó más:

```
db.poblacion.aggregate(
    [
        {
            $match: {
                "2017T": { $gte: 300000 }
            }
        },
        {
            $count: "total"
        }
    ]
)
```

Explicación:
- Filtra los documentos con población total >= 300.000 en el año 2017.
- Obtiene el número de documentos.

Resultado:

```
{ "total" : 12 }
```

SQL:

```
SELECT COUNT(1) AS total
  FROM poblacion
 WHERE '2017T' >= 300000;
```

$limit (LIMIT, sólo en algunos RDBMS)

Deja pasar a la siguiente etapa de la secuencia de agregación únicamente los n primeros documentos, siendo n un número especificado.

Sintaxis:

```
{
    $limit: <positive integer>
}
```

Ejemplo:

Obtener la provincia, el municipio y la población total de los 5 primeros municipios de la lista que se obtiene al seleccionar aquellos que en el año 2017 tenían 300.000 habitantes ó más.

```
db.poblacion.aggregate(
    [
        {
            $match: {
                "2017T": { $gte: 300000 }
            }
        },
        {
            $project: {
                NOM_PROVINCIA:1,
                NOM_MUNICIPIO:1,
                '2017T':1,
                _id:0
            }
        },
        {
            $limit: 5
        }
    ]
)
```

Explicación:
- Filtra los documentos con población total >= 300.000 en el año 2017.
- Selecciona los campos: NOM_PROVINCIA, NOM_MUNICIPIO y '2017T'. Omite _id.
- Selecciona los 5 primeros documentos.

Resultado:

```
{ "NOM_PROVINCIA" : "Palmas, Las", "NOM_MUNICIPIO" : "Palmas de Gran Canaria (Las)", "2017T" : 377650 }
{ "NOM_PROVINCIA" : "Alicante/Alacant", "NOM_MUNICIPIO" : "Alicante/Alacant", "2017T" : 329988 }
{ "NOM_PROVINCIA" : "Balears, Illes", "NOM_MUNICIPIO" : "Palma de Mallorca", "2017T" : 406492 }
{ "NOM_PROVINCIA" : "Barcelona", "NOM_MUNICIPIO" : "Barcelona", "2017T" : 1620809 }
{ "NOM_PROVINCIA" : "Valencia/València", "NOM_MUNICIPIO" : "Valencia", "2017T" : 787808 }
```

SQL

```
SELECT NOM_PROVINCIA, NOM_MUNICIPIO, '2017T'
  FROM poblacion
 WHERE '2017T' >= 300000
 LIMIT 5;
```

$skip (SKIP, solo en algunos RDBMS)

Omite los primeros n documentos, siendo n un número especificado, y pasa los documentos restantes a la siguiente etapa de agregación.

Sintaxis:

```
{
    $skip: <positive integer>
}
```

Ejemplo:

```
db.poblacion.aggregate(
    [
        {
            $match: {
                "2017T": { $gte: 300000 }
            }
        },
        {
            $project: {
                NOM_PROVINCIA:1,
                NOM_MUNICIPIO:1,
                '2017T':1,
                _id:0
            }
        },
        {
            $sort: { "2017T":1 }
        },
        {
            $skip: 5
        }
    ]
)
```

Explicación:
- Filtra los documentos con población total >= 300.000 en el año 2017.
- Selecciona los campos: NOM_PROVINCIA, NOM_MUNICIPIO y '2017T'. Oculta _id.
- Ordena los documetos por '2017T' de menor a mayor.
- Descarta los 5 primeros documentos (los de menor población).

Resultado:

```
{ "NOM_PROVINCIA" : "Murcia", "NOM_MUNICIPIO" : "Murcia", "2017T" : 443243 }
{ "NOM_PROVINCIA" : "Málaga", "NOM_MUNICIPIO" : "Málaga", "2017T" : 569002 }
{ "NOM_PROVINCIA" : "Zaragoza", "NOM_MUNICIPIO" : "Zaragoza", "2017T" : 664938 }
{ "NOM_PROVINCIA" : "Sevilla", "NOM_MUNICIPIO" : "Sevilla", "2017T" : 689434 }
{ "NOM_PROVINCIA" : "Valencia/València", "NOM_MUNICIPIO" : "Valencia", "2017T" : 787808 }
{ "NOM_PROVINCIA" : "Barcelona", "NOM_MUNICIPIO" : "Barcelona", "2017T" : 1620809 }
{ "NOM_PROVINCIA" : "Madrid", "NOM_MUNICIPIO" : "Madrid", "2017T" : 3182981 }
```

SQL:

```
  SELECT NOM_PROVINCIA, NOM_MUNICIPIO, '2017T'
```

```
    FROM poblacion
   WHERE '2017T' >= 300000
ORDER BY '2017T' DESC
  SKIP 5;
```

$group (GROUP BY)

Agrupa documentos por una expresión de identificador determinada y, si se especifican, aplica a cada grupo las expresiones de acumulador. Por cada grupo genera un documento que sólo contendrá el campo identificador y, si se especifican, los campos acumulados.

Lista de operadores:

Nombre	Descripción
$sum	Retorna la suma de los valores numéricos, ignorando los no numéricos. Disponible para **$group** y **$project**.
$avg	Retorna la media de los valores numéricos, ignorando los no numéricos. Disponible para **$group** y **$project**.
$first	Retorna los valores del primer documento de cada grupo, según la ordenación que haya sido definida. Disponible sólo para **$group**.
$last	Retorna los valores del último documento de cada grupo, según la ordenación que haya sido definida. Disponible sólo para **$group**.
$max	Retorna el valor más alto de una expresión para cada grupo de documetos. Disponible para **$group** y **$project**.
$min	Retorna el valor más bajo de una expresión para cada grupo de documetos. Disponible para **$group** y **$project**.
$push	Retorna una lista de valores de expresión para cada grupo de documentos. Disponible sólo para **$group**.
$addToSet	Retorna una lista de valores de expresiún únicos para cada grupo de documentos. El orden de los elemenrtos es indeterminado. Disponible sólo **para $group**.
$stdDevPop	Retorna la desviación estándar de población. Disponible para **$group** y **$project**.
$stdDevSamp	Retorna la desviación estándar de la muestra. Disponible para **$group** y **$project**.

Sintaxis:

```
{
    $group: {
        _id: <expression>,
        <field1>: { <accumulator1> : <expression1> },
        ...
    }
```

```
}
```

Ejemplo:

```
db.poblacion.aggregate(
    [
        {
            $group : {
                _id : "$NOM_PROVINCIA",
                municipios: { $push: "$NOM_MUNICIPIO" }
            }
        }
    ]
).pretty()
```

Explicación:
- Obtiene la lista de municipios de cada provincia.

Resultado:

```
{
    "_id" : "Soria",
    "municipios" : [
        "Abejar",
        "Adradas",
        "Ágreda",
        ...
        "Vozmediano",
        "Yanguas",
        "Yelo"
    ]
}
{
    "_id" : "Sevilla",
    "municipios" : [
        "Aguadulce",
        "Alanís",
        "Albaida del Aljarafe",
        ...
        "Cañada Rosal",
        "Isla Mayor",
        "Cuervo de Sevilla (El)"
    ]
}
{
    "_id" : "Melilla",
    "municipios" : [
        "Melilla"
    ]
}
{
    "_id" : "Madrid",
    "municipios" : [
        "Acebeda (La)",
        "Ajalvir",
        "Alameda del Valle",
        ...
```

```
        "Lozoyuela-Navas-Sieteiglesias",
        "Puentes Viejas",
        "Tres Cantos"
    ]
}
{
    "_id" : "Valladolid",
    "municipios" : [
        "Adalia",
        "Aguasal",
        "Aguilar de Campos",
        ...
        "Wamba",
        "Zaratán",
        "Zarza (La)"
    ]
}
{
    "_id" : "Burgos",
    "municipios" : [
        "Villafruela",
        "Villagalijo",
        "Villagonzalo Pedernales",
        ...
        ```

$addFields
Añade nuevos campos a los documentos, pasándolos a la siguiente etapa de la secuencia de agregaci
ón.

Lista de operadoes aritméticos:

Nombre	Descripción
$add	Suma un array de números.
$divide	Divide números.
$mod	Calcula el resto producido al dividir dos números.
$multiply	Multiplica dos números.
$substract	Resta dos números.

Sintaxis:
```

{
$addFields: {
: ,
...
}
}

```
**Ejemplo**:

Obtener la provincia, el municipio, la población total, la diferencia entre hombres y mujeres y e
l % de esta diferencia, para aquellos municipios que en el año 2017 tenían 300.000 habitantes ó m
ás.
```

db.poblacion.aggregate(
```

```
[
{
$match: { "2017T": { $gte: 300000 }
}
},
{
$addFields: { 'codigo': { $add: [ { $multiply: [ 1000, '$COD_PROVINCIA' ] }, '$COD_MUNICIPIO' ] },
'incremento_población': { $subtract: [ '$2017T', '$2016T' ] },
'mas_mujeres': { $subtract: [ '$2017M', '$2017H' ] } } }, { $addFields: {
'tpc_mas_mujeres': { $multiply: [ 10, { $divide: [ '$mas_mujeres', '$2017T' ] } ] }
}
},
{
$project: { 'codigo':1, 'NOM_PROVINCIA': 1, 'NOM_MUNICIPIO': 1, '2017T': 1,
'incremento_población': 1, 'mas_mujeres': 1, 'tpc_mas_mujeres': 1, _id:0 } }, { $sort: { '2017T':-1
}
},
{
$limit: 5
}
]
).pretty()
```

```
**Explicación**:
- Filtra los documentos con población total >= 300.000 en el año 2017.
- Añade tres nuevos campos:
    - nuevo código de municipio, único, teniendo en cuenta el código de provincia: 'codigo'.
    - Incremento de población total respecto al año anterior (2016): 'incremento\_población'.
    - diferencia entre hombres y mujeres: 'mas\_mujeres'.
- Añade un nuevo campo con el % de la diferencia entre hombres y mujeres, al que denomina 'tpv\_m
as\_mujeres'.
- Selecciona los campos: NOM\_PROVINCIA, NOM\_MUNICIPIO, '2017T', 'incremento\_población', mas\_m
ujeres y tpc\_mas\_mujeres. Oculta \_id.
- Ordena los documentos por total de población en sentido descendente.
- Selecciona los 5 primeros documentos.

**Resultado**:
```

```
{
"NOM_PROVINCIA" : "Madrid",
"NOM_MUNICIPIO" : "Madrid",
"2017T" : 3182981,
"codigo" : 28079,
"incremento_población" : 17440,
"mas_mujeres" : 221781,
"tpc_mas_mujeres" : 0.6967713599295755
}
{
"NOM_PROVINCIA" : "Barcelona",
"NOM_MUNICIPIO" : "Barcelona",
"2017T" : 1620809,
```

“codigo” : 8019,
“incremento_población” : 12063,
“mas_mujeres” : 87559,
“tpc_mas_mujeres” : 0.5402178788493894
}
{
“NOM_PROVINCIA” : “Valencia/València”,
“NOM_MUNICIPIO” : “Valencia”,
“2017T” : 787808,
“codigo” : 46250,
“incremento_población” : -2393,
“mas_mujeres” : 37388,
“tpc_mas_mujeres” : 0.4745826394248345
}
{
“NOM_PROVINCIA” : “Sevilla”,
“NOM_MUNICIPIO” : “Sevilla”,
“2017T” : 689434,
“codigo” : 41091,
“incremento_población” : -1132,
“mas_mujeres” : 35110,
“tpc_mas_mujeres” : 0.5092583191429492
}
{
“NOM_PROVINCIA” : “Zaragoza”,
“NOM_MUNICIPIO” : “Zaragoza”,
“2017T” : 664938,
“codigo” : 50297,
“incremento_población” : 3830,
“mas_mujeres” : 28100,
“tpc_mas_mujeres” : 0.4225957908857668
}

```
**SQL**:
```

SELECT COD_PROVINCIA * 1000 + COS_MUNICIPIO AS codigo,
NOM_PROVINCIA,
NOM_MUNICIPIO,
‘2017T’,
as ‘incremento_población’,
‘2017M’ - ‘2017H’ AS mas_mujeres,
10 * (‘2017M’ - ‘2017H’) / ‘2017T’ AS tpc_mas_mujeres
FROM poblacion
WHERE 2017T >= 300000;

```
# $out

Guarda en una nueva colección los documentos resultantes de ejecutar una secuencia de agregación.
 Esta etapa ($out) debe ser la última de la secuencia.
```

Sintaxis:

{
$out: “”
}

Ejemplo:

Guardar en la colección 'madrid2017' el nombre de los municipios de la provincia de Madrid y sus correspondientes cifras de población **del** año 2017. Además, renombrar el nombre **del** dato 'NOM_MUNICIPIO' por 'Muncipio':

db.poblacion.aggregate(
[
{
$match : { COD_PROVINCIA:28 } }, { $project : {
“Municipio”:”$NOM_MUNICIPIO", "2017H":1, "2017M":1, "2017T":1 } }, { $out: “madrid2017”
}
]
)

Explicación:
- Selcciona los documentos de la provincia de código 28.
- Escoge los campos NOM_MUNICIPIO, 2017H, 2017M y 2017T, renombando NOM_MUNICIPIO por Muncipio.
- Guarda los documenros resultantes en la colecciób marid2017.

Resultado:

Ejecutar:

db.madrid2017.find().pretty()

{
“_id” : ObjectId(“5ac771ca9c1000b150eaf7f7”),
“2017T” : 66,
“2017H” : 37,
“2017M” : 29,
“Municipio” : “Acebeda (La)”
}
{
“_id” : ObjectId(“5ac771ca9c1000b150eaf7f8”),
“2017T” : 4455,
“2017H” : 2327,
“2017M” : 2128,
“Municipio” : “Ajalvir”

}
{
“_id” : ObjectId(“5ac771ca9c1000b150eaf7f9”),
“2017T” : 199,
“2017H” : 117,
“2017M” : 82,
“Municipio” : “Alameda del Valle”
}
…
…
…

SQL:

CREATE TABLE madrid2017 as (
Municipio VARCHAR2(100),
‘2017H’ NUMBER(10),
‘2017M’ NUMBER(10),
‘2017T’ NUMBER(10)
);

INSERT INTO madrid2017
SELECT NOM_MUNICIPIO as Municipio, ‘2017H’, ‘2017M’, ‘2017T’
FROM poblacion
WHERE COD_PROVINCIA=28;

 # $bucket
Agrupa los documentos entrantes en segmentos, en función de una expresión y unos límites específicos.

Sintaxis:

{
$bucket: { groupBy: <expression>, boundaries: [<lowerbound1>, <lowerbound2>, ...], default: <literal>, output: { <output1>: { <$accumulator expression> },
…
: { <$accumulator expression> }
}
}
}

Ejemplo:

Agrupar en los segmentos que se especifican a continuación, los municipios de la provincia de Madrid según su número total de habitantes en el año 2017:
- de 0 a 100
- de 100 a 1.000
- de 1.000 a 10.000
- de 10.000 a 50.000
- de 50.000 a 100.000

```
  - el resto (más de 100.000)
```

db.poblacion.aggregate(
[
{
$match: { COD_PROVINCIA: 28 } }, { $bucket: {
groupBy: ‘$2017T', boundaries: [0, 100, 1000, 10000, 50000, 100000], default: 'Más de 100.000',
output: { 'Total': { $sum: 1 },
‘Municipios’: { $push: '$NOM_MUNICIPIO’ }
}
}
}
]
).pretty()

```
  **Explicación**:
  - Filtra los documentos con código de provincia 28 (Madrid).
  - Clasifica los municipios según su población total en el año 2017, haciendo grupos con los valor
  es límites 0, 100, 1.000, 10.000, 50.000, 100.000 y más de 100.000.
  - Por cada grupo muestra su identificador (\_id), el número total de municipios y la lista de mun
  icipios.

  **Resultado**:
```

{
“_id” : 0,
“Total” : 10,
“Municipios” : [
“Acebeda (La)”,
“Atazar (El)”,
“Hiruela (La)”,
“Horcajuelo de la Sierra”,
“Madarcos”,
“Puebla de la Sierra”,
“Robledillo de la Jara”,
“Robregordo”,
“Serna del Monte (La)”,
“Somosierra”
]
}
{
“_id” : 100,
“Total” : 37,
“Municipios” : [
“Alameda del Valle”,
“Ambite”,
“Berzosa del Lozoya”,
“Berrueco (El)”,
“Braojos”,

“Brea de Tajo”,
“Cabanillas de la Sierra”,
“Canencia”,
“Cervera de Buitrago”,
“Corpa”,
“Garganta de los Montes”,
“Gargantilla del Lozoya y Pinilla de Buitrago”,
“Gascones”,
“Horcajo de la Sierra-Aoslos”,
“Lozoya”,
“Montejo de la Sierra”,
“Navarredonda y San Mamés”,
“Olmeda de las Fuentes”,
“Patones”,
“Pezuela de las Torres”,
“Pinilla del Valle”,
“Piñuécar-Gandullas”,
“Prádena del Rincón”,
“Redueña”,
“Ribatejada”,
“Rozas de Puerto Real”,
“Santorcaz”,
“Torremocha de Jarama”,
“Valdaracete”,
“Valdemanco”,
“Valdelaguna”,
“Valdemaqueda”,
“Valdepiélagos”,
“Valverde de Alcalá”,
“Villamanrique de Tajo”,
“Villavieja del Lozoya”,
“Puentes Viejas”
]
}
{
“_id” : 1000,
“Total” : 81,
“Municipios” : [
“Ajalvir”,
“Álamo (El)”,
“Aldea del Fresno”,
“Anchuelo”,
“Batres”,
“Becerril de la Sierra”,
“Belmonte de Tajo”,
“Boalo (El)”,
“Buitrago del Lozoya”,
“Bustarviejo”,
“Cabrera (La)”,
“Camarma de Esteruelas”,
“Cadalso de los Vidrios”,

“Campo Real”,
“Carabaña”,
“Casarrubuelos”,
“Cenicientos”,
“Cercedilla”,
“Cobeña”,
“Colmenar del Arroyo”,
“Colmenar de Oreja”,
“Colmenarejo”,
“Collado Mediano”,
“Cubas de la Sagra”,
“Chinchón”,
“Chapinería”,
“Estremera”,
“Fresnedillas de la Oliva”,
“Fresno de Torote”,
“Fuente el Saz de Jarama”,
“Fuentidueña de Tajo”,
“Guadalix de la Sierra”,
“Hoyo de Manzanares”,
“Loeches”,
“Manzanares el Real”,
“Miraflores de la Sierra”,
“Molar (El)”,
“Molinos (Los)”,
“Moraleja de Enmedio”,
“Morata de Tajuña”,
“Navacerrada”,
“Navalafuente”,
“Navalagamella”,
“Navas del Rey”,
“Nuevo Baztán”,
“Orusco de Tajuña”,
“Pedrezuela”,
“Pelayos de la Presa”,
“Perales de Tajuña”,
“Pozuelo del Rey”,
“Quijorna”,
“Rascafría”,
“Robledo de Chavela”,
“San Martín de Valdeiglesias”,
“Santa María de la Alameda”,
“Santos de la Humosa (Los)”,
“Serranillos del Valle”,
“Sevilla la Nueva”,
“Soto del Real”,
“Talamanca de Jarama”,
“Tielmes”,
“Titulcia”,
“Torrejón de la Calzada”,
“Torrejón de Velasco”,

“Torrelaguna”,
“Torres de la Alameda”,
“Valdeavero”,
“Valdeolmos-Alalpardo”,
“Valdetorres de Jarama”,
“Valdilecha”,
“Vellón (El)”,
“Venturada”,
“Villaconejos”,
“Villa del Prado”,
“Villamanta”,
“Villamantilla”,
“Villanueva de Perales”,
“Villar del Olmo”,
“Villarejo de Salvanés”,
“Zarzalejo”,
“Lozoyuela-Navas-Sieteiglesias”
]
}
{
“_id” : 10000,
“Total” : 29,
“Municipios” : [
“Algete”,
“Alpedrete”,
“Arroyomolinos”,
“Brunete”,
“Ciempozuelos”,
“Colmenar Viejo”,
“Daganzo de Arriba”,
“Escorial (El)”,
“Galapagar”,
“Griñón”,
“Guadarrama”,
“Humanes de Madrid”,
“Meco”,
“Mejorada del Campo”,
“Moralzarzal”,
“Navalcarnero”,
“Paracuellos de Jarama”,
“San Agustín del Guadalix”,
“San Fernando de Henares”,
“San Lorenzo de El Escorial”,
“San Martín de la Vega”,
“Torrelodones”,
“Valdemorillo”,
“Velilla de San Antonio”,
“Villalbilla”,
“Villanueva de la Cañada”,
“Villanueva del Pardillo”,
“Villaviciosa de Odón”,

“Tres Cantos”
]
}
{
“_id” : 50000,
“Total” : 12,
“Municipios” : [
“Aranjuez”,
“Boadilla del Monte”,
“Arganda del Rey”,
“Collado Villalba”,
“Coslada”,
“Majadahonda”,
“Pinto”,
“Pozuelo de Alarcón”,
“Rivas-Vaciamadrid”,
“Rozas de Madrid (Las)”,
“San Sebastián de los Reyes”,
“Valdemoro”
]
}
{
“_id” : “Más de 100.000”,
“Total” : 10,
“Municipios” : [
“Alcalá de Henares”,
“Alcobendas”,
“Alcorcón”,
“Fuenlabrada”,
“Getafe”,
“Leganés”,
“Madrid”,
“Móstoles”,
“Parla”,
“Torrejón de Ardoz”
]
}

```
# $bucketAuto
Agrupa los documentos entrantes en segmentos, en función de una expresión y un número total de gr
upos. Los límites de cada segmento o grupo se determinan automaticamente intentando distribuir lo
s documentos de manera uniforme.

**Sintaxis**:
```

{
$bucketAuto: { groupBy: <expression>, buckets: <number>, output: { <output1>: {
<$accumulator expression> },
...
}
Granularity:

}
}

Ejemplo:

Desitribuir en 10 grupos los municipios de la provincia de Madrid según su número total de habitantes en el año 2017:

```
db.poblacion.aggregate(
[
{
$match: { COD_PROVINCIA: 28 } }, { $bucketAuto: {
groupBy: “$2017T”,
buckets: 10
}
}
]
).pretty()
```

Explicación:
- Filtra los documentos con código de provincia 28 (Madrid).
- Clasifica los municipios según su población total en el año 2017, distribuyéndolos en 10 grupos, intentando que contengan cada uno de ellos el mismo número de integrantes.
- Por cada grupo muestra los valores mínimo y máximo (de población) y el total de elementos (municipios).

Resultado:

```
{ “_id” : { “min” : 44, “max” : 203 }, “count” : 18 }
{ “_id” : { “min” : 203, “max” : 684 }, “count” : 18 }
{ “_id” : { “min” : 684, “max” : 1340 }, “count” : 18 }
{ “_id” : { “min” : 1340, “max” : 2225 }, “count” : 18 }
{ “_id” : { “min” : 2225, “max” : 3906 }, “count” : 18 }
{ “_id” : { “min” : 3906, “max” : 6424 }, “count” : 18 }
{ “_id” : { “min” : 6424, “max” : 9093 }, “count” : 18 }
{ “_id” : { “min” : 9093, “max” : 20320 }, “count” : 18 }
{ “_id” : { “min” : 20320, “max” : 71299 }, “count” : 18 }
{ “_id” : { “min” : 71299, “max” : 3182981 }, “count” : 17 }
```

 # $unwind
Desconstruye un campo de matriz de los documentos de entrada para generar un documento para cada elemento. Cada documento de salida reemplaza la matriz con un valor de elemento. Para cada documento de entrada, genera n documentos donde n es la cantidad de elementos de la matriz y puede ser cero para una matriz vacía.

Sintaxis:

```
{
```

$unwind:
}

Ejemplo:

De la consulta anterior como ejemplo de '$bucket', seleccionar el documento que contenía los municipios de entre 10.000 y 50.000 habitantes y generar un nuevo documento por cada uno de esos municipios.

db.poblacion.aggregate(
[
{
$match: { COD_PROVINCIA: 28 } }, { $bucket: {
groupBy: '$2017T', boundaries: [0, 100, 1000, 10000, 50000, 100000], default: 'Más de 100.000',
output: { 'Total': { $sum: 1 },
'Municipios': { $push: '$NOM_MUNICIPIO' }
}
}
},
{
$match: { _id: 50000 } }, { $unwind: '$Municipios'
}
]
).pretty()

Explicación:
- Filtra los documentos con código de provincia 28 (Madrid).
- Clasifica los municipios según su población total en el año 2017, haciendo grupos con los valores límites 0, 100, 1.000, 10.000, 50.000, 100.000 y más de 100.000.
- Por cada grupo muestra su identificador (_id), el número total de municipios y la lista de municipios.
- Seleccina el documento con _id: 50000
- Crea un nuevo documento por cada elemento de la matriz 'Municipios' del documento anterior.

Resultado:

{ "_id" : 50000, "Total" : 12, "Municipios" : "Aranjuez" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Boadilla del Monte" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Arganda del Rey" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Collado Villalba" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Coslada" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Majadahonda" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Pinto" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Pozuelo de Alarcón" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Rivas-Vaciamadrid" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "Rozas de Madrid (Las)" }
{ "_id" : 50000, "Total" : 12, "Municipios" : "San Sebastián de los Reyes" }

```
# $sample

Selecciona aleatoriamente n documentos, siendo n un número especificado, y los pasa a la siguient
e etapa de agregación.

**Sintaxis**:
```

{
$sample: { size: }
}

```
**Ejemplo**:

Mostrar el nombre de provincia, nopmbre de municipio y población total de 5 municipios al hazar q
ue en el año 2017 tenían 300.000 habitantes ó más:
```

db.poblacion.aggregate(
[
{
$match: { "2017T": { $gte: 300000 }
}
},
{
$sample: { size: 5 } }, { $project: {
NOM_PROVINCIA:1,
NOM_MUNICIPIO:1,
‘2017T’:1,
_id:0
}
}
]
)

```
**Explicación**:
- Filtra los documentos con población total >= 300.000 en el año 2017.
- Toma 5 de ellos al azar.
- Selecciona los campos: NOM\_PROVINCIA, NOM\_MUNICIPIO y '2017T'. Oculta \_id.

**Resultado**:
```

{ “NOM_PROVINCIA” : “Málaga”, “NOM_MUNICIPIO” : “Málaga”, “2017T” : 569002 }
{ “NOM_PROVINCIA” : “Barcelona”, “NOM_MUNICIPIO” : “Barcelona”, “2017T” : 1620809 }
{ “NOM_PROVINCIA” : “Valencia/València”, “NOM_MUNICIPIO” : “Valencia”, “2017T” : 787808 }
{ “NOM_PROVINCIA” : “Bizkaia”, “NOM_MUNICIPIO” : “Bilbao”, “2017T” : 345110 }

```
# $collStats
Devuelve estadísticas con respecto a una colección o vista.

**Sintaxis**:
```

{
$collStats: {
latencyStats: { histograms: },
StorageStats: {},
Count: {}
}
}

Ejemplo:

db.poblacion.aggregate(
[
{
$collStats: { latencyStats: { histograms: true } }
}
]
).pretty()

Resultado:

{
“ns” : “test.poblacion”,
“localTime” : ISODate(“2018-04-09T09:07:05.294Z”),
“latencyStats” : {
“reads” : {
“histogram” : [
{
“micros” : NumberLong(32),
“count” : NumberLong(2)
},
{
“micros” : NumberLong(64),
“count” : NumberLong(10)
},
{
“micros” : NumberLong(128),
“count” : NumberLong(8)
},
{
“micros” : NumberLong(512),
“count” : NumberLong(2)
},
{
“micros” : NumberLong(4096),
“count” : NumberLong(1)
},
{
“micros” : NumberLong(6144),
“count” : NumberLong(10)

},
{
“micros” : NumberLong(8192),
“count” : NumberLong(4)
},
{
“micros” : NumberLong(16384),
“count” : NumberLong(1)
}
],
“latency” : NumberLong(132379),
“ops” : NumberLong(38)
},
“writes” : {
“histogram” : [ ],
“latency” : NumberLong(0),
“ops” : NumberLong(0)
},
“commands” : {
“histogram” : [ ],
“latency” : NumberLong(0),
“ops” : NumberLong(0)
}
}
}

```
# $indexStats
Devuelve estadísticas sobre el uso de cada índice para la colección.

**Sintaxis**:
```

{
$indexStats: { }
}

```
**Ejemplo**:
```

db.poblacion.aggregate(
[
{
$indexStats: { }
}
]
).pretty()

```
**Resultado**:
```

{

“name” : “NOM_MUNICIPIO_1”,
“key” : {
“NOM_MUNICIPIO” : 1
},
“host” : “r2d2:27017”,
“accesses” : {
“ops” : NumberLong(0),
“since” : ISODate(“2018-04-09T10:24:39.238Z”)
}
}
{
“name” : “*id*”,
“key” : {
“_id” : 1
},
“host” : “r2d2:27017”,
“accesses” : {
“ops” : NumberLong(0),
“since” : ISODate(“2018-04-09T10:24:28.640Z”)
}
}
{
“name” : “COD_PROVINCIA_1_COD_MUNICIPIO_1”,
“key” : {
“COD_PROVINCIA” : 1,
“COD_MUNICIPIO” : 1
},
“host” : “r2d2:27017”,
“accesses” : {
“ops” : NumberLong(0),
“since” : ISODate(“2018-04-09T10:24:28.650Z”)
}
}

```
---------------------------------------------------------------------

# $currentOp
Devuelve información sobre las operaciones activas y/o inactivas para la implementación de MongoD
B.

**Sintaxis**:
```

{
$currentOp: {
allUsers: , idleConnections:
}
}

```
# $facet
Procesa varias interconexiones de agregación en una sola etapa en el mismo conjunto de documentos
```

de entrada. Permite la creación de agregaciones multifacéticas capaces de caracterizar datos en múltiples dimensiones, o facetas, en una sola etapa.

Sintaxis:

{
$facet: {
: [, , ...],
: [, , ...],
...
}
}

 # $geoNear
Devuelve una secuencia ordenada de documentos en función de la proximidad a un punto geoespacial. Incorpora la funcionalidad de $match, $sort y $limit para datos geoespaciales. Los documentos de salida incluyen un campo de distancia adicional y pueden incluir un campo de identificación de ubicación.

Sintaxis:

{
$geoNear: {

}
}

 # $graphLookup
Realiza una búsqueda recursiva en una colección. Para cada documento de salida agrega un nuevo campo de matriz que contiene los resultados transversales de la búsqueda recursiva de ese documento.

Sintaxis:

{
$graphLookup: {
from: ,
startWith: ,
connectFromField: ,
connectToField: ,
as: ,
maxDepth: ,
depthField: ,
restrictSearchWithMatch:
}
}

 # $listLocalSessions
Enumera todas las sesiones activas recientemente en uso en la instancia de mongos o mongod actual

mente conectada. Es posible que estas sesiones aún no se hayan propagado a la colección system.se
ssions.

Sintaxis:

{
$listLocalSessions:
}

$listSessions
Enumera todas las sesiones que han estado activas el tiempo suficiente para propagarse a la colec
ción system.sessions.

Sintaxis:

{
$listSessions:
}

$lookup (JOIN)
Realiza una combinación externa izquierda a otra colección en la misma base de datos para filtrar
documentos de la colección "unida" para su procesamiento.

Sintaxis:

{
$lookup: {
from: ,
localField: ,
foreignField: ,
as:
}
}

$redact
Cambia la forma de cada documento en la secuencia al restringir el contenido de cada documento se
gún la información almacenada en los documentos. Incorpora la funcionalidad de $project y $match.
Se puede usar para implementar la redacción a nivel de campo. Para cada documento de entrada, ge
nera uno o cero documentos.

Sintaxis:

{
$redact:
}

$replaceRoot
Reemplaza un documento con el documento incrustado especificado. La operación reemplaza todos los

```
 campos existentes en el documento de entrada, incluido el campo _id. Especifique un documento in
crustado en el documento de entrada para promover el documento incrustado al nivel superior.

**Sintaxis**:
```

{
$replaceRoot: {
newRoot:
}
}

```
# $sortByCount
Agrupa los documentos entrantes según el valor de una expresión específica y luego calcula el rec
uento de documentos en cada grupo distinto.

**Sintaxis**:
```

{
$sortByCount:
}
```

# Referencias


https://docs.mongodb.com/manual/aggregation/
https://docs.mongodb.com/manual/reference/sql-aggregation-comparison/
https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/
https://docs.mongodb.com/manual/reference/operator/aggregation/
```