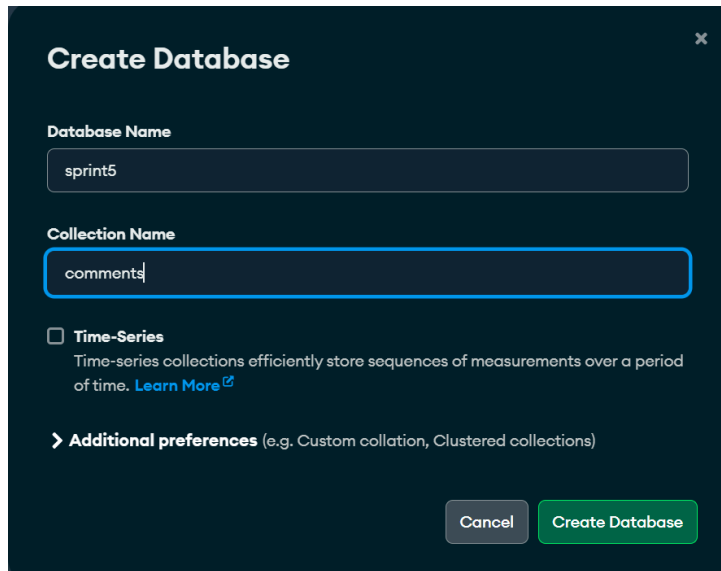


SPRINT 5: Consultes amb MongoDB

Nivell 1

Crea una base de dades amb MongoDB utilitzant com a col·leccions els arxius adjunts.



Create Database

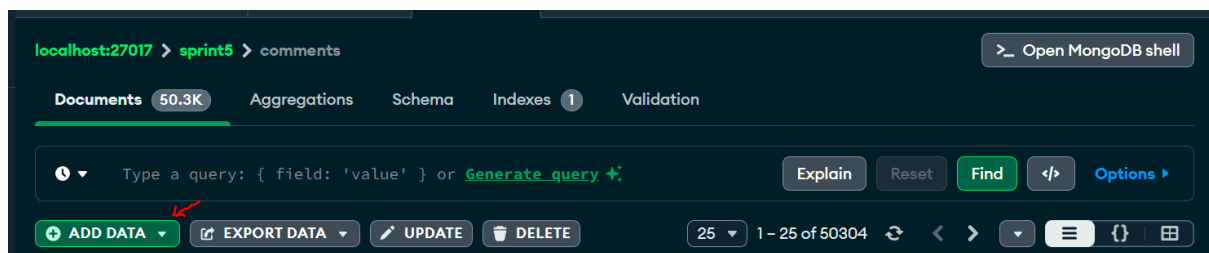
Database Name
sprint5

Collection Name
comments

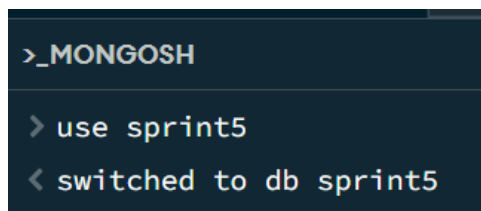
☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

> **Additional preferences** (e.g. Custom collation, Clustered collections)

Cancel Create Database



Per aquesta entrega hem d'instal·lar la eina mongoDB. Una vegada instal·lada amb el compass podem crear la nostra base de dades (que anomenarem sprint5) i anirem afegint col·leccions (una per cada arxiu JSON que conté les dades per fer l'exercici). Després al shell li demanarem que utilitzi la base de dades que hem creat mitjançant: **use sprint5**



```
> _MONGOSH
> use sprint5
< switched to db sprint5
```

EXERCICI 1 - NIVELL 1

- Mostra els 2 primers comentaris que hi ha en la base de dades

```
>_MONGOSH
> db.comments.find({}, { text: 1, _id: 0}).limit(2)
< {
  text: 'Rem officiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates neque culpa molestias. Ut
}
{
  text: 'Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Commodi nisi sit placeat rerum vero cupid
}
```

Per poder mostrar els dos primers comentaris, hem de seleccionar la col·lecció que contingui els comentaris dels usuaris. Amb la funció `db.comments.find()`, seleccionem la base de dades `comments` i li demanem que ens trobi dades segons les condicions que volguem especificar dins el parèntesi. Aquesta funció realment retorna un cursor, és a dir, un objecte que representa el conjunt dels resultats de la consulta, sense ser tenir el contingut d'aquesta. Si no afegíssim cap condició dins el parèntesi, ens tornaria un objecte que representaria tots els camps dels 20 primers resultats de la col·lecció.

En aquest cas, però, dins el parèntesi tenim una primera condició `{}` que actua com un `SELECT *` a SQL. Aquest agafaria tots els camps disponibles al objecte que estem demanant.

No obstant, dins el parèntesi de `find` estem especificant quins camps en concret volem que es mostrin. Amb `text: 1` estem demanant que el camp `text` (que és el cos del comentari) es mostri. Amb `_id: 0` estem demanant que no es mostri el camp `_id`. Quan especifiquem que mostri un camp com `camp: 1` ja estem exclouent tots els altres camps, però per defecte, el camp `_id` es mostra sempre. Llavors, necessitem especificar `_id: 0` perquè no aparegui.

- Quants usuaris tenim registrats?

```
> db.users.find().count()
< 185
```

Com hem comentat anteriorment, demanarem amb la funció `find` que ens trobi les dades de `users`. A continuació afegim una altra funció seguida d'un punt, en aquest cas `count()`. Aquesta compta el nombre de documents que retorna la funció anterior (`find()`).

- Quants cinemes hi ha en l'estat de Califòrnia?

```
> db.theaters.find({"location.address.state": "CA"}).count()  
< 169
```

De nou amb la funció `find`, en aquest cas seleccionant la col·lecció `theaters` per referir-nos als cinemes, afegirem la condició `location.address.state: "CA"` per filtrar només pels cinemes de l'estat de Califòrnia. En quant a la sintàxi per referir-nos al camp `state`, ho fem seguit de punts perquè es tracta d'un objecte anidat dins el camp `location`. Per accedir-hi haurem de seleccionar els camps de més a fora cap a dins seguits de punts. Per últim, fora de la funció `find`, afegirem un `count()` per comptar el nombre de documents retornats filtrats pel `find`. Tenim un total de 169 cinemes a Califòrnia.

- Quin va ser el primer usuari/ària en registrar-se?

```
> db.comments.find({}, {name: 1, _id: 0}).sort({date: 1}).limit(1)  
< {  
  name: 'Mercedes Tyler'  
}
```

Mitjançant `find`, en aquest cas de la col·lecció `comments`, filtrem pel nom: 1 i `_id`: 0. Per altra banda, també farem una funció `sort`. Aquesta funció s'assimila al que seria un `ORDER BY` a SQL. D'aquesta manera, ordenem pel camp `date: 1`, això ordena els resultats de les dates d'enregistrament de manera ascendent, tenint el primer resultat com el més antic. Per últim només ens cal encadenar les funcions amb un `.limit(1)`, que agafarà només el primer resultat. Si volguéssim mostrar tots els camps disponibles en el cursor només caldria eliminar la condició `{name: 1, _id: 0}` de la query. Trobem que Mercedes Tyler és l'usuari més antic en registrar-se.

- Quantes pel·lícules de comèdia hi ha en la nostra base de dades?

```
> db.movies.find({genres: "Comedy"}).count()  
< 7024
```

En aquesta query farem un `find`, filrant pel gènere comèdia ("Comedy") i comptant quants resultats trobem. Tenim un total de 7024 pel·lícules de comèdia.

EXERCICI 2

- Mostra'm tots els documents de les pel·lícules produïdes en 1932, però que el gènere sigui drama o estiguin en francès.

```
> db.movies.find({
  $or: [
    {genres: "drama"},
    {languages: "French"}],
  year: 1932})
< {
  _id: ObjectId('573a1391f29313caabcd9458'),
  plot: 'A young artist draws a face at a canvas on his easel. Suddenly the mouth on the drawing comes into life and st
  runtime: 55,
  rated: 'UNRATED',
  cast: [
    'Enrique Rivero',
    'Elizabeth Lee Miller',
    'Pauline Carton',
    'Odette Talazac'
  ],
  num_mflix_comments: 1,
  poster: 'https://m.media-amazon.com/images/M/MV5BYWY3ODE5ZWEtYjlmYi00NjA4LTk4ZWYtMzBhZDE5MjY0YTYxXkEyXkFqcGdeQXVyNzI=
  title: 'The Blood of a Poet',
  lastupdated: '2015-09-16 13:13:05.537000000',
  languages: [
    'French'
```

En aquest exercici, tornem a demanar a la base de dades que ens trobi informació amb `find`. En aquest cas, dins el `find` ens demanen tres condicions. La primera condició és obligatòria i és que la pel·lícula hagi estat produïda l'any 1932. Després ens dona dues condicions que poden estar una o l'altra, sense dependre de la coexistència de les dues. És a dir, que el gènere sigui drama o l'idioma sigui en francès.

Per fer-ho haurem de filtrar la consulta amb una condició `$or: [{genres: "drama"}, {languages: "French"}]` aquesta condició fa una operació lògica OR dins d'un *array* d'expressions i selecciona els documents que satisfan almenys alguna de les dues condicions que hem posat (o bé el gènere drama o bé l'idioma francès).

Per últim, i fora del *array*, afegim una altra condició (encara estem dins el parèntesi del `find`). Afegim la condició obligatòria que hem comentat abans: que la pel·lícula sigui de l'any 1932.

EXERCICI 3

- Mostra'm tots els documents de pel·lícules estatunidenques que tinguin entre 5 i 9 premis que van ser produïdes entre 2012 i 2014.

```
> db.movies.find({
  "awards.wins": {$gte: 5, $lte: 9},
  year: {$gte: 2012, $lte: 2014},
  countries: "USA"})
< {
  _id: ObjectId('573a13acf29313caabd29366'),
  fullplot: "The manager of the negative assets sector of Life magazine, Walter Mitty, has been working for sixteen year:
  imdb: {
    rating: 7.4,
    votes: 211230,
    id: 359950
  },
  year: 2013,
  plot: 'When his job along with that of his co-worker are threatened, Walter takes action in the real world embarking on
  genres: [
    'Adventure',
    'Comedy',
    'Drama'
  ],
  rated: 'PG',
  metacritic: 54,
  title: 'The Secret Life of Walter Mitty',
  lastupdated: '2015-08-31 00:10:51.747000000'.
```

En aquest exercici introduïm dos noves funcions. L'enunciat ens demana que filtrem per pel·lícules estatunidenques que es continguin entre dos valors de premis i anys. Per trobar resultats entre dos valors a SQL, ho fariem amb la funció BETWEEN. Ara però, ho fem amb les funcions \$gte i \$lte, dins de la funció find. \$gte filtra els resultats majors o iguals (>=) al número que li demanem. \$lte filtra pels resultats menors o iguals. Per tant, repetirem aquestes funcions pels dos objectes que volem filtrar (*awards.wins* i *year*). Recordem que *wins* es troba anidat dins d'*awards* i per tant per cridar-lo haurem d'escriure-ho seguit d'un punt. Per últim filtrem pel país → *countries*: "USA".

EXERCICI 1 - NIVELL 2

- Compte quants comentaris escriu un usuari/ària que utilitza "GAMEOFTHRON.ES" com a domini de correu electrònic.

```
> db.comments.find({ "email": /gameofthron\.es$/ }).count()
< 22841
```

En aquest exercici, filtrem per l'*email* de manera que acabi amb el domini gameofthron.es. Per fer-ho ho fem amb una funció que s'anomena *regex* que permet a la nostra query buscar cadenes de text que coincideixin amb el que li demanem a la query.

Per fer-ho, posem el text *gameofthron* entre barres horitzontals que indiquen que es tracta d'una expressió regular de text. La última part *.es* ha de sortir de les barres horitzontals,

perquè es pot interpretar dins la funció com una cerca de qualsevol caràcter. El símbol \$ del final ens assegura que tanquem l'expressió i que no busca valors més enllà del .es final.

EXERCICI 2

- Quants cinemes hi ha en cada codi postal situats dins de l'estat Washington D. C. (DC)?

```
> db.theaters.aggregate([
  {$match: {"location.address.state": "DC"}},
  {$group: {
    _id: "$location.address.zipcode",
    count: { $sum: 1 }}
  ]})
< {
  _id: '20016',
  count: 1
}
{
  _id: '20010',
  count: 1
}
{
  _id: '20002',
  count: 1
}
```

Aquest exercici es pot resoldre mitjançant un pipeline d'agregació. Aquest ens permet fer càlculs i agrupacions una mica més complexes de manera seguida, de tal forma que el resultat de la segona part del pipeline dependrà del primer.

Així doncs, la primera part conté una funció \$match filtra per l'anidació de location.address.state, fent que mostri els documents que pertanyen a l'estat de Washington D.C.

A continuació, mitjançant \$group, agrupa per codi postal aquells documents que ja han estat filtrats per estat. Per últim, mitjançant count i \$sum: 1, sumarem un valor per cada cinema que trobi per codi postal. De totes maneres, només hi ha 3 codis postals i hi ha un cinema per cada.

EXERCICI 1 - NIVELL 3

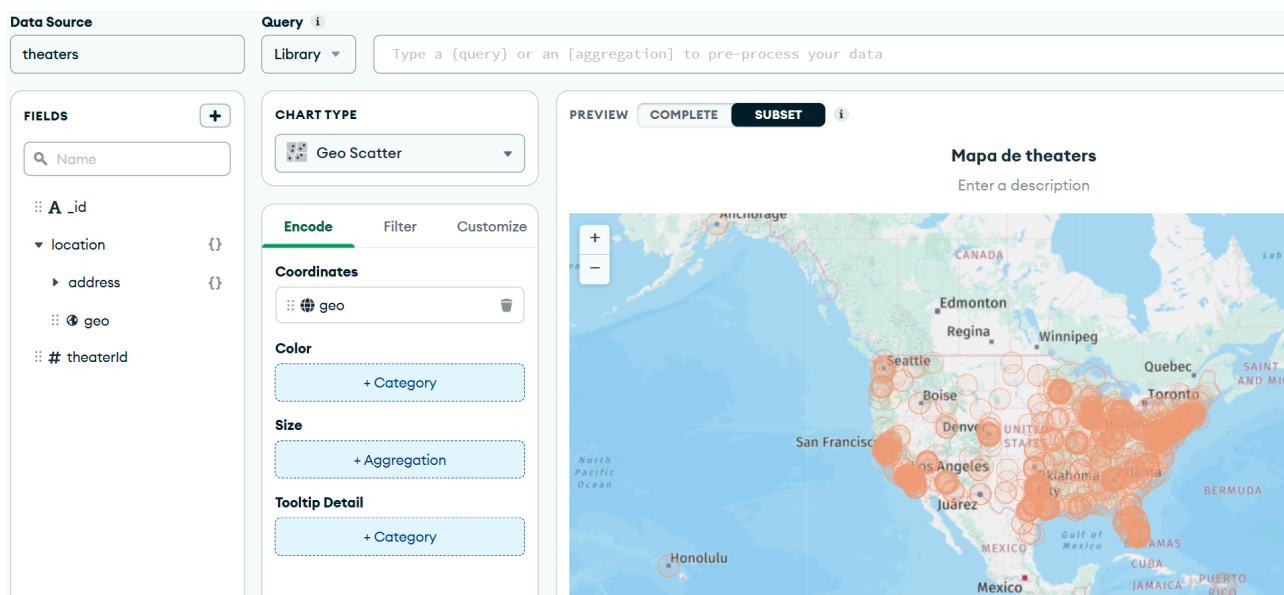
- Troba totes les pel·lícules dirigides per John Landis amb una puntuació IMDb (Internet Movie Database) d'entre 7,5 i 8.

```
> db.movies.find({
  "imdb.rating": {
    $gte: 7.5, $lte: 8},
  directors: "John Landis"})
< {
  _id: ObjectId('573a1397f29313caabce6d94'),
  fullplot: "Faber College has one frat house so disreputable it will take anyone. It has a second one full of white, an
  imdb: {
    rating: 7.6,
    votes: 84834,
    id: 77975
  },
  year: 1978,
  plot: 'At a 1962 college, Dean Vernon Wormer is determined to expel the entire Delta Tau Chi Fraternity, but those troi
  genres: [
    'Comedy'
  ],
  rated: 'R',
  metacritic: 82,
  title: 'Animal House',
  lastupdated: '2015-09-13 00:02:47.803000000',
  languages: [
    'English',
```

En aquest exercici, filtrem per la col·lecció de pel·lícules i pel *rating* d'*imdb* afegint les clàusules *\$gte* i *\$lte* per ajustar la nota mitjana de 7,5 a 8 (inclosos) i filtrant pel director “John Landis”

EXERCICI 2

- Mostra en un mapa la ubicació de tots els teatres de la base de dades.



En aquest últim exercici necessitem utilitzar una eina anomenada MongoDB Atlas. Amb aquesta podem pujar les dades geogràfiques (en coordenades) de les nostres dades i amb un mapa tipus geo scatter veurem on es troben tots els cinemes de la col·lecció “theaters”.