

Taller : MongoDB con JSON de Pokémon

Conexión mujeres Tic

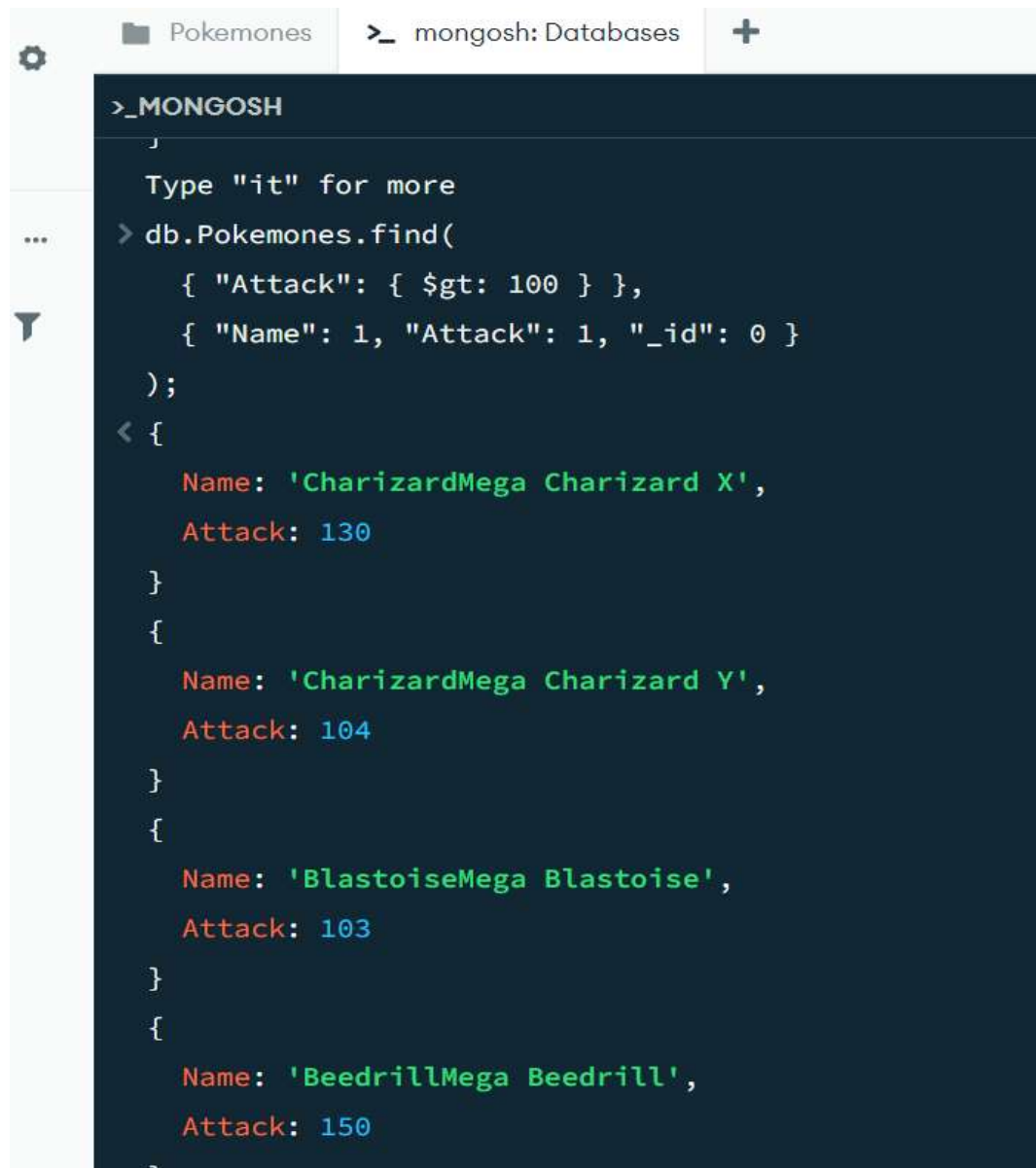
Tina Paola Varela Monzón

1. Encuentra todos los Pokémon de tipo "Electric".

```
>_MONGOSH
> use Pokemon
< switched to db Pokemon
> db.Pokemones.aggregate([
  { $match: { Type1: "Electric" } }
]);
< {
  _id: ObjectId('67eddfc5e14fb6422178349c'),
  Num: 25,
  Name: 'Pikachu',
  Type1: 'Electric',
  HP: 35,
  Attack: 55,
  Defense: 40,
  SpAtk: 50,
  SpDef: 50,
  Speed: 90,
  Generation: 1,
  Legendary: false
}
```

```
>_MONGOSH
  _id: ObjectId('67eddfc5e14fb6422178349d'),
  Num: 26,
  Name: 'Raichu',
  Type1: 'Electric',
  HP: 60,
  Attack: 90,
  Defense: 55,
  SpAtk: 90,
  SpDef: 80,
  Speed: 110,
  Generation: 1,
  Legendary: false
}
{
  _id: ObjectId('67eddfc5e14fb642217834d6'),
  Num: 81,
  Name: 'Magnezone',
  Type1: 'Electric',
  Type2: 'Steel',
  HP: 25,
  Attack: 35,
  Defense: 70,
  SpAtk: 95,
```

2. Muestra solo los nombres y el ataque de los Pokémon con más de 100 de ataque.



The screenshot shows a MongoDB CLI window with the following content:

```
>_MONGOSH
Type "it" for more
> db.Pokemones.find(
  { "Attack": { $gt: 100 } },
  { "Name": 1, "Attack": 1, "_id": 0 }
);
< {
  Name: 'CharizardMega Charizard X',
  Attack: 130
}
{
  Name: 'CharizardMega Charizard Y',
  Attack: 104
}
{
  Name: 'BlastoiseMega Blastoise',
  Attack: 103
}
{
  Name: 'BeedrillMega Beedrill',
  Attack: 150
}
```

The interface includes a top bar with a folder icon labeled 'Pokemones', a terminal icon labeled 'mongosh: Databases', and a plus sign. On the left side, there is a sidebar with a gear icon, an ellipsis, and a funnel icon.

3. Encuentra los Pokémon cuya defensa esté entre 80 y 100 (inclusive).

```
>_MONGOSH
> db.Pokemones.find(
  { "Attack": { $gt: 100 } },
  { "Name": 1, "Attack": 1, "_id": 0 }
);
< {
  Name: 'CharizardMega Charizard X',
  Attack: 130
}
{
  Name: 'CharizardMega Charizard Y',
  Attack: 104
}
{
  Name: 'BlastoiseMega Blastoise',
  Attack: 103
}
{
  Name: 'BeedrillMega Beedrill',
  Attack: 150
}
{
  Name: 'Nidoking',
  Attack: 102
}
```

>_MONGOSH

```
  Name: 'BeedrillMega Beedrill',  
  Attack: 150  
}  
{  
  Name: 'Nidoking',  
  Attack: 102  
}  
{  
  Name: 'Primeape',  
  Attack: 105  
}  
{  
  Name: 'Arcanine',  
  Attack: 110  
}  
{  
  Name: 'Machamp',  
  Attack: 130  
}  
{  
  Name: 'Victreebel',  
  Attack: 105
```

4. Muestra el promedio de ataque de los Pokémon por tipo (Type1).

```
>_MONGOSH
type: 10 for more
> db.Pokemones.aggregate([
  {
    $group: {
      _id: "$Type1",
      promedioAtaque: { $avg: "$Attack" }
    }
  }
]);
< {
  _id: 'Fairy',
  promedioAtaque: 61.529411764705884
}
{
  _id: 'Flying',
  promedioAtaque: 78.75
}
{
  _id: 'Fire',
  promedioAtaque: 84.76923076923077
}
{
  _id: 'Grass',
```

5. Encuentra el Pokémon con más HP de cada tipo.

```
> _MONGOSH
}
> db.Pokemones.aggregate([
  {
    $sort: { HP: -1 }
  },
  {
    $group: {
      _id: "$Type1",
      pokemonConMasHP: { $first: "$$ROOT" }
    }
  },
  {
    $project: {
      tipo: "$_id",
      nombre: "$pokemonConMasHP.Name",
      hp: "$pokemonConMasHP.HP",
      _id: 0
    }
  }
]);
< {
  tipo: 'Water',
```


>_MONGOSH

```
< {  
  tipo: 'Water',  
  nombre: 'Wailord',  
  hp: 170  
}  
{  
  tipo: 'Flying',  
  nombre: 'Noivern',  
  hp: 85  
}  
{  
  tipo: 'Grass',  
  nombre: 'Gogoat',  
  hp: 123  
}  
{  
  tipo: 'Fire',  
  nombre: 'Entei',  
  hp: 115  
}  
{  
  tipo: 'Normal',  
  nombre: 'Blissey'
```

6. Muestra los 5 Pokémon más rápidos.

```
>_MONGOSH
> db.Pokemones.find().sort({ "Speed": -1 }).limit(5);
< {
  _id: ObjectId('67eddfc5e14fb6422178362d'),
  Num: 386,
  Name: 'DeoxysSpeed Forme',
  Type1: 'Psychic',
  HP: 50,
  Attack: 95,
  Defense: 90,
  SpAtk: 95,
  SpDef: 90,
  Speed: 180,
  Generation: 3,
  Legendary: true
}
{
  _id: ObjectId('67eddfc5e14fb642217835b9'),
  Num: 291,
  Name: 'Ninjask',
  Type1: 'Bug',
  Type2: 'Flying',
  HP: 61,
  Attack: 90,
  Defense: 45,
  SpAtk: 50,
  SpDef: 50,
  Speed: 160,
```

Parte 3: Combinaciones de \$match, \$group y \$sort

7.  Muestra el promedio de ataque de los Pokémon tipo “Water” ordenado de mayor a menor.


```
>_MONGOSH
}
> db.Pokemones.find().sort({ "Speed": -1 }).limit(5);
< {
  _id: ObjectId('67eddfc5e14fb6422178362d'),
  Num: 386,
  Name: 'DeoxysSpeed Forme',
  Type1: 'Psychic',
  HP: 50,
  Attack: 95,
  Defense: 90,
  SpAtk: 95,
  SpDef: 90,
  Speed: 180,
  Generation: 3,
  Legendary: true
}
{
  _id: ObjectId('67eddfc5e14fb642217835b9'),
  Num: 291,
  Name: 'Ninjask',
  Type1: 'Bug',
  Type2: 'Flying',
  HP: 61,
  Attack: 90,
  Defense: 45,
  SpAtk: 50,
  SpDef: 50,
  Speed: 160,
```

8. Encuentra el Pokémon con más ataque por generación y ordénalos de mayor a menor.

```
>_MONGOSH
  Legendary: true
}
> db.Pokemones.aggregate([
  {
    $sort: { "Attack": -1 }
  },
  {
    $group: {
      _id: "$Generation",
      pokemonConMasAtaque: { $first: "$$ROOT" }
    }
  },
  {
    $project: {
      generacion: "$_id",
      nombre: "$pokemonConMasAtaque.Name",
      ataque: "$pokemonConMasAtaque.Attack",
      _id: 0
    }
  },
  {
    $sort: { ataque: -1 }
  }
]);
< {
  generacion: 1,
  nombre: 'MewtwoMega Mewtwo X',
  ataque: 190
```

>_MONGOSH

```
    generacion: 1,
    nombre: 'MewtwoMega Mewtwo X',
    ataque: 190
  }
  {
    generacion: 2,
    nombre: 'HeracrossMega Heracross',
    ataque: 185
  }
  {
    generacion: 3,
    nombre: 'RayquazaMega Rayquaza',
    ataque: 180
  }
  {
    generacion: 5,
    nombre: 'KyuremBlack Kyurem',
    ataque: 170
  }
  {
    generacion: 4,
    nombre: 'GarchompMega Garchomp',
    ataque: 170
  }
  {
    generacion: 6,
    nombre: 'DiancieMega Diancie',
    ataque: 160
  }
```

9.  Crea un índice en el campo Type1.

```
    ataque: 160
  }
> db.Pokemones.createIndex({ "Type1": 1 });
< Type1_1
Pokemon >
```

10. Usa explain() para analizar el rendimiento de una búsqueda:

```
>_MONGOSH
< Type1_1
> db.Pokemones.find({ "Type1": "Fire" }).explain("executionStats");
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Pokemon.Pokemones',
    parsedQuery: {
      Type1: {
        '$eq': 'Fire'
      }
    },
  },
  indexFilterSet: false,
  queryHash: '135BE0C6',
  planCacheShapeHash: '135BE0C6',
  planCacheKey: '3B166ACA',
  optimizationTimeMillis: 3,
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
  prunedSimilarIndexes: false,
  winningPlan: {
    isCached: false,
    stage: 'FETCH',
    inputStage: {
      stage: 'IXSCAN',
      keyPattern: {
        Type1: 1
      }
    }
  }
}
```

```

indexFilterSet: false,
queryHash: '135BE0C6',
planCacheShapeHash: '135BE0C6',
planCacheKey: '3B166ACA',
optimizationTimeMillis: 3,
maxIndexedOrSolutionsReached: false,
maxIndexedAndSolutionsReached: false,
maxScansToExplodeReached: false,
prunedSimilarIndexes: false,
winningPlan: {
  isCached: false,
  stage: 'FETCH',
  inputStage: {
    stage: 'IXSCAN',
    keyPattern: {
      Type1: 1
    },
    indexName: 'Type1_1',
    isMultiKey: false,
    multiKeyPaths: {
      Type1: []
    },
    isUnique: false,
    isSparse: false,
    isPartial: false,
    indexVersion: 2,
    direction: 'forward',
    indexBounds: {

```

11. Crea un índice compuesto en Type1 y Speed, y analiza una búsqueda:

```
}  
> db.Pokemones.createIndex({ "Type1": 1 });  
< Type1_1  
> db.Pokemones.find({ "Type1": "Fire" }).explain("executionStats");  
< {  
  explainVersion: '1',  
  queryPlanner: {  
    namespace: 'Pokemon.Pokemones',  
    parsedQuery: {  
      Type1: {  
        '$eq': 'Fire'  
      }  
    },  
    indexFilterSet: false,  
    queryHash: '135BE0C6',  
    planCacheShapeHash: '135BE0C6',  
    planCacheKey: '3B166ACA',  
    optimizationTimeMillis: 3,  
    maxIndexedOrSolutionsReached: false,  
    maxIndexedAndSolutionsReached: false,  
    maxScansToExplodeReached: false,  
    prunedSimilarIndexes: false,  
    winningPlan: {  
      isCached: false,  
      stage: 'FETCH',  
      inputStage: {
```

```

>_MONGOSH
prunedSimilarIndexes: false,
winningPlan: {
  isCached: false,
  stage: 'FETCH',
  inputStage: {
    stage: 'IXSCAN',
    keyPattern: {
      Type1: 1
    },
    indexName: 'Type1_1',
    isMultiKey: false,
    multiKeyPaths: {
      Type1: []
    },
    isUnique: false,
    isSparse: false,
    isPartial: false,
    indexVersion: 2,
    direction: 'forward',
    indexBounds: {
      Type1: [
        '["Fire", "Fire"]'
      ]
    }
  }
},
rejectedPlans: [
  {

```

En las consultas del taller, usé `find()` para las consultas simples (#1, #2, #3, #6) y `aggregate()` para operaciones más complejas que requieren agrupación, proyección o múltiples etapas (#4, #5, #7, #8).

Por ejemplo, en la consulta #6 (los 5 Pokémon más rápidos), podríamos usar tanto `find()` como `aggregate()`: