

◆ Parte 1: Consultas Básicas

1. 🧑 Encuentra todos los Pokémon de tipo “Electric”.

```
>_MONGOSH
> // Contar
fireCount=db.Pokemones.countDocuments({$or: [{Type1: "Fire"}, {Type2: "Fire"}]}); 

// Mostrar el conteo
print(`Total de Pokémon tipo Fire: ${fireCount}`);

// Obtener listado
db.Pokemones.find({$or: [{Type1: "Fire"}, {Type2: "Fire"}]}, {_id: 0, Name: 1, Type1: 1, Type2: 1});
< Total de Pokémon tipo Fire: 64
< [
  {
    Name: 'Charmander',
    Type1: 'Fire'
  }
  {
    Name: 'Charmeleon',
    Type1: 'Fire'
  }
  {
    Name: 'Charizard',
    Type1: 'Fire',
    Type2: 'Flying'
  }
]
>_MONGOSH
> [
  {
    Name: 'CharizardMega Charizard X',
    Type1: 'Fire',
    Type2: 'Dragon'
  }
  {
    Name: 'CharizardMega Charizard Y',
    Type1: 'Fire',
    Type2: 'Flying'
  }
  {
    Name: 'Vulpix',
    Type1: 'Fire'
  }
  {
    Name: 'Ninetales',
    Type1: 'Fire'
  }
  {
    Name: 'Growlithe',
    Type1: 'Fire'
  }
]
```

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

```
>_MONGOSH
> attackCount=db.Pokemones.countDocuments({ Attack: { $gt: 100 } });
// Mostrar el conteo
print(`Total de Pokémon con Attack mayor que 100: ${attackCount}`);
db.Pokemones.find(
  { Attack: { $gt: 100 } },
  { _id: 0, Name: 1, Attack: 1 }
).sort({ Attack: -1 });
< Total de Pokémon con Attack mayor que 100: 170
< [
  {
    Name: 'MewtwoMega Mewtwo X',
    Attack: 190
  },
  {
    Name: 'HeracrossMega Heracross',
    Attack: 185
  },
  {
    Name: 'RayquazaMega Rayquaza',
    Attack: 180
  },
  {
    Name: 'DeoxysAttack Forme',
    Attack: 180
  }
]
```

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

```
>_MONGOSH
> db.Pokemones.find(
  { Defense: { $gte: 80, $lte: 100 } },
  { _id: 0, Name: 1, Defense: 1 }
).sort({ Defense: -1 });
< [
  {
    Name: 'Pelipper',
    Defense: 100
  },
  {
    Name: 'Victini',
    Defense: 100
  },
  {
    Name: 'Blastoise',
    Defense: 100
  },
  {
    Name: 'ShayminLand Forme',
    Defense: 100
  },
  {
    Name: 'Slaking',
    Defense: 100
  }
]
```

```
>_MONGOSH
  Defense: 100
}
{
  Name: 'Metang',
  Defense: 100
}
{
  Name: 'Whirlipede',
  Defense: 99
}
{
  Name: 'Cradily',
  Defense: 97
}
{
  Name: 'Dragonite',
  Defense: 95
}
{
  Name: 'Magneton',
  Defense: 95
}
{
```

Parte 2: Agregaciones

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

```
>_MONGOSH
> db.Pokemones.aggregate([
  {$group:{_id:"$Type1", avgAttack:{$avg:"$Attack"}}, },
  {
    $project:{Type:"$_id", avgAttack:{$round:[ "$avgAttack",2]}, _id:0},
    {$sort:{avgAttack:-1}}
  ]));
< [
  {
    Type: 'Dragon',
    avgAttack: 112.12
  },
  {
    Type: 'Fighting',
    avgAttack: 96.78
  },
  {
    Type: 'Ground',
    avgAttack: 95.75
  },
  {
    Type: 'Rock',
    avgAttack: 92.86
  }
]
```

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

```
>_MONGOSH

Pokemon > db.Pokemones.aggregate([
    {$sort:{HP:-1}}, //Ordenar todos los Pokémon por HP de mayor a menor
    {$group:{_id: "$Type1", //Agrupar por Type1
        maxHP:{$max:"$HP"}, //Obtener el máximo HP del grupo
        pokemonConMaxHP:{$first:"$$ROOT"}}, //Guardar documento del Pokémon que tiene mayor HP
    },
    {$replaceRoot:{newRoot:"$pokemonConMaxHP"}}, //Reemplazar la raíz del documento por "pokemonConMaxHP"
    {
        $project: { //Proyectar solo los campos deseados
            _id: 0,
            Name: 1,
            Type1: 1,
            HP: 1
        }
    },
    {$sort:{Type1:1}} //Ordenar los resultados por Type1 (A-Z)
])
```



```
>_MONGOSH

{
    Name: 'Yanmega',
    Type1: 'Bug',
    HP: 86
}
{
    Name: 'Yveltal',
    Type1: 'Dark',
    HP: 126
}
{
    Name: 'KyuremBlack Kyurem',
    Type1: 'Dragon',
    HP: 125
}
{
    Name: 'Ampharos',
    Type1: 'Electric',
    HP: 90
}
{
    Name: 'Xerneas',
    Type1: 'Fairy',
    HP: 126
}
```

6.  Muestra los 5 Pokémon más rápidos.

```
>_MONGOSH
}
> db.Pokemones.find({}, {_id:0, Name:1, Speed:1}).sort({Speed:-1}).limit(5);
< [
  {
    Name: 'DeoxysSpeed Forme',
    Speed: 180
  },
  {
    Name: 'Ninjask',
    Speed: 160
  },
  {
    Name: 'AlakazamMega Alakazam',
    Speed: 150
  },
  {
    Name: 'DeoxysAttack Forme',
    Speed: 150
  },
  {
    Name: 'DeoxysNormal Forme',
    Speed: 150
  }
]
Pokemon >
```

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
Pokemon > db.Pokemones.aggregate([
  {$match:{Type1: "Water"}}, //Filtrar solo los Pokémon de tipo "Water" en Type1
  {
    $group:{_id: null, //Agrupa todos los documentos filtrados (sin separar por otro campo)
            avgAttack:{$avg:"$Attack"} //Calcula el promedio de "Attack"
    },
    {
      $project:{_id: 0, //Excluye el campo "_id"
                "Tipo Water": "Water", //Muestra el tipo como "Water"
                "Promedio de Ataque":{$round:[{"$avgAttack",2]}} //Redondea a 2 decimales
      },
      {$sort:{ "Promedio de Ataque": -1}} //Ordena (aunque solo habrá un resultado)
    }
]);
```

```
< [
  {
    'Tipo Water': 'Water',
    'Promedio de Ataque': 74.15
  }
]
Pokemon > |
```

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

```
>_MONGOSH
Pokemon> db.Pokemones.aggregate([
    {$group:{_id:"$Generation", //Agrupamiento por generación
        MaxAttack:{$max:"$Attack"}},
    },
    {$lookup:{ //Busca los Pokémon que tengan ese ataque máximo en esa generación
        from:"Pokemones",
        let:{gen:"$_id", atk:"$MaxAttack"}, //Define variables para usar en el pipeline
        pipeline:[
            //El pipeline es una cadena de etapas que permiten transformar los datos paso a paso.
            //Es muy útil para realizar análisis complejos dentro de MongoDB, sin necesidad de
            //procesar los datos fuera de la base de datos. Es especialmente poderoso en el
            //operador $aggregate, pero también aparece dentro de $lookup y otros contextos.
            {$match:{$expr:{ //Expresión que permite usar variables con $let
                $and:[{$eq:[{"$Generation", "$gen"]}, {"$eq:[{"$Attack", "$atk"}]}]] //coinciden generación y ataque máximo
            }},
            {$project:{_id:0, Name:1, Attack:1, Generation:1}},
            ], as: "TopAttacker"}, //El resultado del lookup se almacena en este campo
            {$unwind:"$TopAttacker"}, //Descompone el array "TopAttacker" en documentos individuales
            {$replaceRoot:{newRoot:"$TopAttacker"}}, //Reemplaza la raíz del documento actual con el contenido de "TopAttacker"
            {$sort:{Attack:-1}} //Ordena los resultados por valor de ataque, de mayor a menor
        ]});
]);
< {
    Name: 'MewtwoMega Mewtwo X',
    Attack: 190,
    Generation: 1
}
{
    Name: 'HeracrossMega Heracross',
    Attack: 185,
    Generation: 2
}
{
    Name: 'GroudonPrimal Groudon',
    Attack: 180,
    Generation: 3
}
{
    Name: 'RayquazaMega Rayquaza',
    Attack: 180,
    Generation: 3
}
{
    Name: 'DeoxysAttack Forme',
```

Parte 4: Indexación y Rendimiento

9.  Crea un índice en el campo Type1.

```
>_MONGOSH
> db.Pokemones.createIndex({ Type1: 1 })
< Type1_1
Pokemon> |
```

```
>_MONGOSH
> db.Pokemones.createIndex({ Type1: 1 })
< Type1_1
> db.Pokemones.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { Type1: 1 }, name: 'Type1_1' }
]
Pokemon> |
```

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

```
>_MONGOSH
>
> db.Pokemones.find({ Type1: "Fire" }).explain("executionStats")
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Pokemon.Pokemones',
    parsedQuery: {
      Type1: {
        '$eq': 'Fire'
      }
    },
    indexFilterSet: false,
    queryHash: '135BE0C6',
    planCacheShapeHash: '135BE0C6',
    planCacheKey: 'F89FA229',
    optimizationTimeMillis: 9,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
    }
  }
}
```

```
>_MONGOSH
{
  '$db': 'Pokemon',
},
serverInfo: {
  host: 'CMT-RosarioPenuela',
  port: 27017,
  version: '8.0.6',
  gitVersion: '80f21521ad4a3dfd5613f5d649d7058c6d46277f'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}
Pokemon> |
```

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

```
>_MONGOSH
> //Crear el índice compuesto
db.Pokemones.createIndex({ Type1: 1, Speed: -1 })
< Type1_1_Speed_-1
> //Realizar una búsqueda y analizarla con explain()
db.Pokemones.find({
  Type1: "Fire",
  Speed: { $gt: 80 }
}).explain("executionStats")
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Pokemon.Pokemones',
    parsedQuery: {
      '$and': [
        {
          Type1: {
            '$eq': 'Fire'
          }
        },
        {
          Speed: {
            '$gt': 80
          }
        }
      ]
    }
  }
}
```

```
>_MONGOSH
{
  '$db': 'Pokemon'
},
serverInfo: {
  host: 'CMT-RosarioPenuela',
  port: 27017,
  version: '8.0.6',
  gitVersion: '80f21521ad4a3dfd5613f5d649d7058c6d46277f'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}
Pokemon> |
```

Referencias

```
>_MONGOSH

> //Crear el índice compuesto
db.Pokemones.createIndex({ Type1: 1, Speed: -1 })
< Type1_1_Speed_-1
> //Realizar una búsqueda y analizarla con explain()
db.Pokemones.find({
  Type1: "Fire",
  Speed: { $gt: 80 }
}).explain("executionStats")
< {
  explainVersion: '1',
  queryPlanner: {
    namespace: 'Pokemon.Pokemones',
    parsedQuery: {
      '$and': [
        {
          Type1: {
            '$eq': 'Fire'
          }
        },
        {
          Speed: {
            '$gt': 80
          }
        }
      ]
    }
  }
}
```

```
>_MONGOSH
    type1: 1,
    Speed: -1
},
indexName: 'Type1_1_Speed_-1',
isMultiKey: false,
multiKeyPaths: {
  Type1: [],
  Speed: []
},
isUnique: false,
isSparse: false,
isPartial: false,
indexVersion: 2,
direction: 'forward',
indexBounds: {
  Type1: [
    '["Fire", "Fire"]'
  ],
  Speed: [
    '[inf.0, 80)'
  ]
}
```