¿Qué es la Agregación?

La agregación en MongoDB es el proceso mediante el cual transformamos, agrupamos y analizamos datos para obtener información significativa. Es similar a GROUP BY, COUNT, SUM, etc., en SQL, pero mucho más poderosa.

🔧 ¿Qué es una Aggregation Pipeline?

La tubería de agregación (aggregation pipeline) es una secuencia de etapas por donde pasan los documentos. Cada etapa transforma los datos y pasa el resultado a la siguiente.

Sintaxis:

db.coleccion.aggregate([

{ <etapa\_1> },

{ <etapa\_2> },

...

]);

🔄 Etapas más comunes

**1. $match – Filtrar documentos (como find())**

db.estudiantes.aggregate([

{ $match: { edad: { $lt: 20 } } }

]);

**2. $project – Seleccionar o crear nuevos campos**

db.estudiantes.aggregate([

{ $project: { nombre: 1, edad: 1, \_id: 0 } }

]);

Crear alias o combinar campos:

{ $project: { nombreCompleto: { $concat: ["$nombre", " ", "$apellido"] } } }

**3. $group – Agrupar por un campo (clave) y aplicar funciones agregadas**

db.estudiantes.aggregate([

{ $group: { \_id: "$programa", cantidad: { $sum: 1 } } }

]);

Más funciones: $sum, $avg, $min, $max, $push, $addToSet

**4. $sort – Ordenar**

{ $sort: { edad: 1 } } // 1: ascendente, -1: descendente

**5. $limit – Limitar la cantidad de resultados**

{ $limit: 3 }

**6. $skip – Saltar los primeros n resultados**

{ $skip: 2 }

**7. $count – Contar resultados**

{ $count: "total" }

**8. $sortByCount – Contar y ordenar por frecuencia de un campo**

{ $sortByCount: "$programa" }

**9. $unwind – Descomponer arrays (útil para arrays de subdocumentos)**

{ $unwind: "$tareas" }

=============================================================================

🎓 Ejemplos Aplicados

📘 Librería – Libros publicados después de 2016 y agrupados por género:

db.libros.aggregate([

{ $match: { publicacion: { $gt: ISODate("2016-01-01") } } },

{ $group: { \_id: "$genero", totalLibros: { $sum: 1 } } }]);

🎬 Cine – Películas agrupadas por director y promedio de nota IMDb:

db.peliculas.aggregate([

{ $unwind: "$puntuaciones" },

{ $match: { "puntuaciones.medio": "IMDb" } },

{ $group: {

\_id: "$director",

promedioIMDb: { $avg: "$puntuaciones.nota" }

} }

]);

🧑‍💼 Empresa – Cantidad de tareas distintas por empleado activo:

db.empleados.aggregate([

{ $match: { activo: true } },

{ $project: { nombre: 1, cantidadTareas: { $size: "$tareas" }, \_id: 0 } }

]);

=============================================================================

💡 Tips para practicar

Comenzá con $match para filtrar.

Sumá $project si necesitás ver campos específicos.

Probá agrupar con $group.

Combiná etapas: match → group → sort → limit.

Usá $unwind si hay arrays como tareas, críticas o puntuaciones.

**GUÍA DE PRÁCTICA DE AGREGACIÓN EN MONGODB**

**1. $match – Filtrar documentos**

**Ejercicio 1.1:** Mostrar libros con más de 300 páginas.

db.libros.aggregate([{ $match: { paginas: { $gt: 300 } } }]);

**Ejercicio 1.2:** Mostrar empleados activos.

db.empleados.aggregate([{ $match: { activo: true } }]);

**2. $project – Seleccionar o crear nuevos campos**

**Ejercicio 2.1:** Mostrar título y ciudad de la editorial, sin el \_id.

db.libros.aggregate([{ $project: { titulos: 1, "editorial.ciudad": 1, \_id: 0 } }]);

**Ejercicio 2.2:** Mostrar nombre completo del empleado (nombre apellido).

db.empleados.aggregate([{$project: {\_id: 0, nombreCompleto: { $concat: ["$nombre", " ", "$apellido"] } }}]);

**3. $group – Agrupar y aplicar funciones agregadas**

**Ejercicio 3.1:** Agrupar películas por director y contar cuántas hizo cada uno.

db.peliculas.aggregate([{ $group: { \_id: "$director", cantidadPeliculas: { $sum: 1 } } }]);

**Ejercicio 3.2:** Agrupar empleados según si están activos o no, y contar cada grupo.

db.empleados.aggregate([ { $group: { \_id: "$activo", total: { $sum: 1 } } }]);

**4. $sort – Ordenar resultados**

**Ejercicio 4.1:** Mostrar películas ordenadas por fecha de estreno (más nuevas primero).

db.peliculas.aggregate([ { $sort: { estreno: -1 } }]);

**Ejercicio 4.2:** Ordenar libros por número de páginas de menor a mayor.

db.libros.aggregate([{ $sort: { paginas: 1 } }]);

**5. $limit – Limitar la cantidad de documentos**

**Ejercicio 5.1:** Mostrar los 3 empleados más antiguos (por fecha de ingreso).

db.empleados.aggregate([{ $sort: { ingreso: 1 } }, { $limit: 3 }]);

**Ejercicio 5.2:** Mostrar los primeros 2 libros con más páginas.

db.libros.aggregate([{ $sort: { paginas: -1 } },{ $limit: 2 }]);

**6. $skip – Saltar documentos**

**Ejercicio 6.1:** Saltar los primeros 2 empleados y mostrar los siguientes 2.

db.empleados.aggregate([{ $sort: { nombre: 1 } },{ $skip: 2 },

{ $limit: 2 }]);

**Ejercicio 6.2:** Saltar el libro más nuevo y mostrar los siguientes 3 por fecha.

db.libros.aggregate([{ $sort: { publicacion: -1 } },{ $skip: 1 },

{ $limit: 3 }]);

**7. $count – Contar resultados**

**Ejercicio 7.1:** Contar películas ganadoras de un Oscar.

db.peliculas.aggregate([{ $match: { "premios.oscar": true } },

{ $count: "ganadorasOscar" }]);

**Ejercicio 7.2:** Contar libros con precio efectivo menor a $4000.

db.libros.aggregate([{ $match: { "precios.tipo": "efectivo", "precios.precio": { $lt: 4000 } } },{ $count: "librosBaratos" }]);

**8. $sortByCount – Agrupar, contar y ordenar**

**Ejercicio 8.1:** Contar y ordenar libros por género.

db.libros.aggregate([{ $sortByCount: "$genero" }]);

**Ejercicio 8.2:** Contar y ordenar empleados por cantidad de tareas.

db.empleados.aggregate([{ $unwind: "$tareas" },{ $sortByCount: "$tareas" }]);

**9. $unwind – Descomponer arrays**

**Ejercicio 9.1:** Mostrar cada tarea individual de cada empleado.

db.empleados.aggregate([{ $unwind: "$tareas" },

{ $project: { nombre: 1, tarea: "$tareas", \_id: 0 } }]);

**Ejercicio 9.2:** Mostrar críticas individuales de cada película.

db.peliculas.aggregate([

{ $unwind: "$criticas" },

{ $project: { titulo: 1, autor: "$criticas.autor", comentario: "$criticas.comentario", \_id: 0 } }

]);

**EJERCICIOS COMBINADOS (Desafío)**

**Desafío 1: Mostrar los nombres de los 3 empleados activos con más tareas.**

db.empleados.aggregate([

{ $match: { activo: true } },

{ $project: { nombre: 1, cantidadTareas: { $size: "$tareas" } } },

{ $sort: { cantidadTareas: -1 } },

{ $limit: 3 }

]);

**Desafío 2: Agrupar libros por ciudad de editorial y mostrar solo los que tengan más de 1 libro.**

db.libros.aggregate([

{ $group: { \_id: "$editorial.ciudad", total: { $sum: 1 } } },

{ $match: { total: { $gt: 1 } } }

]);