Group con más de un campo

Nuestra base de datos:

Hemos creado una base de datos diseñada para llevar un registro de una imaginaria tienda de informática.

Esta tienda adquiere sus productos de determinados proveedores y los guarda en sus distintos almacenes. Cuando les llega un pedido simplemente se ocupan de enviar el producto al comprador.

Componentes de nuestra base de datos:

Esta sería la estructura de los datos que insertamos.

- El campo **"_id"** actuaría de clave primaria de nuestra base de datos al ser un valor único para cada documento.
- El documento anidado **"detallesVenta"** contendría información propia de la venta como "**fechaVenta"**, **"fechaEntrega"**, **"almacen"**, **"descuento"** o **"unidades"**, algunos de estos campos no los utilizaremos para nuestra práctica pero son necesarios para que la base de datos sea lógica con su propósito teórico (llevar un registro de ventas).
- El documento anidado "**detallesProducto"** contendría información del producto que vamos a vender, como el **"proveedor"** que nos ha proveído esos productos de la venta, el mismo **"producto"** que vamos a vende y su **"costeUnidad"** que es lo que cuesta su compra (ya que no es fabricado por la empresa que lo vende) y su **"precioUnidad"** que como su nombre indica es por lo que lo vendemos.

```
Estructura:
    _id
    detallesVenta: {
        fechaVenta: Date
        fechaEntrega: Date
        almacen: String
        descuento: Int
        unidades: Int
    }
    detallesProducto: {
        proveedor: String
        producto: String
        costeUnidad: Int
        precioUnidad: Int
    }
```

Funcionamiento de un Aggregate:

El Aggregate es una herramienta que nos permite modificar la información original de nuestra base de datos mediante una serie de etapas.

Es clave para entender el funcionamiento del Aggregate comprender que cada etapa solamente puede trabajar con la información que ha dejado la etapa anterior, es decir, si yo en mi primera etapa he desechado un campo “Cantidad”, en la etapa actual no podré utilizarlo.

Cada etapa tiene su propio uso:

- **$match:** Selecciona todos los documentos que cumplan el requisito que le impongamos, los documentos no seleccionados serán desechados como expliqué arriba.
- **$group**: Nos permite reestructurar toda nuestra información respecto a uno o varios campos _id, es decir, podemos llegar a esta etapa con 15 documentos y mediante esta nueva agrupación o reorganización nos podríamos quedar entre 1 y 15 documentos.
- **$project:** Nos permite decidir qué campos queremos suprimir o incluso crear nuevos campos.
- **$sort:** Nos permite ordenar toda la información que obtengamos de nuestra sentencia.

Nuestra sentencia Aggregate:

Este sería nuestro Aggregate completo, lo explicaré etapa por etapa para una mejor comprensión.

A rasgos generales he decidido crear un aggregate que nos permita estudiar el rendimiento que han tenido los distintos Almacenes a lo largo de los meses que han tenido compras.

```
db.ordenadores.aggregate([
    {$match:
        {"detallesProducto.proveedor": "Jaimexo"}
    },
    {$group:
        {
            _id: {
                Almacén:"$detallesVenta.almacen",
                Mes: {$month: "$detallesVenta.fechaVenta"}
            },
            gananciasTotales: {$sum: {$multiply: ["$detallesVenta.unidades","$detallesProducto.precioUnidad",{$sum: ["$detallesVenta.descuento",1]}]}},
            costesTotales: {$sum: {$multiply: ["$detallesVenta.unidades","$detallesProducto.costeUnidad"]}}
        }
    },
    {$project:
        {
            Almacén: "$_id.Almacén",
            Mes: "$_id.Mes",
            _id: 0,
            gananciasNeto: "$gananciasTotales",
            costesTotales: "$costesTotales",
            beneficioNeto: {$round: {$subtract: ["$gananciasTotales","$costesTotales"]}},
            impuestos: {$round: {$multiply: [{$subtract: ["$gananciasTotales","$costesTotales"]},0.21]}},
            beneficioTotal: {$round: {$multiply: [{$subtract: ["$gananciasTotales","$costesTotales"]},1.21]}}
        }
    },
    {$sort:
        {"Almacén":1, "Mes":1}
    },
    {$match:
        {$expr: {$lt: [{$multiply: ["$costesTotales",1.5]},"$beneficioTotal"]}}
    }
]).pretty();
```

1) Primer Match

Este primer match selecciona todos los documentos de nuestra base de datos original cuyo proveedor sea “Jaimexo”.

```
    {$match:
        {"detallesProducto.proveedor": "Jaimexo"}
    },
```

2) Group

Este group toma como campos **_id** los campos **Almacén** y **Mes** creando así una serie de documentos como los de la segunda captura.

También agrega dos campos a partir de los campos de la base de datos original; **“gananciasTotales”** y **“costesTotales”** cuya creación es necesaria para poder trabajar con ellos en la siguiente etapa.

```
{$group:
    {
        _id: {
            Almacén:"$detallesVenta.almacen",
            Mes: {$month: "$detallesVenta.fechaVenta"}
        },
        gananciasTotales: {$sum: {$multiply: ["$detallesVenta.unidades","$detallesProducto.precioUnidad",{$sum: ["$detallesVenta.descuento",1]}]}},
        costesTotales: {$sum: {$multiply: ["$detallesVenta.unidades","$detallesProducto.costeUnidad"]}}
    }
},
```

```
{ "_id" : { "Almacén" : "Barcelona", "Mes" : 8 } }
{ "_id" : { "Almacén" : "Barcelona", "Mes" : 7 } }
{ "_id" : { "Almacén" : "Madrid", "Mes" : 5 } }
{ "_id" : { "Almacén" : "Madrid", "Mes" : 6 } }
{ "_id" : { "Almacén" : "Madrid", "Mes" : 4 } }
{ "_id" : { "Almacén" : "Sevilla", "Mes" : 3 } }
{ "_id" : { "Almacén" : "Barcelona", "Mes" : 10 } }
{ "_id" : { "Almacén" : "Barcelona", "Mes" : 3 } }
{ "_id" : { "Almacén" : "Madrid", "Mes" : 1 } }
>
```

3) Project

Este project en primer lugar modifica los que en la etapa anterior eran campos clave permitiéndonos una mayor legibilidad de nuestros resultados.

En lugar de presentarnos **“Almacén”** y **“Mes”** como documentos hijos de **“_id”** los torna como campos normales a pesar de que siguen siendo respecto a los cuales existen el resto de campos.

También edita el nombres de uno de los campos creados anteriormente a **“gananciasNeto”** mientras que **“costesTotales”** sigue igual.

En último lugar crea los campos **“beneficioNeto”**, **“impuestos”** y **“beneficioTotal”**, a los cuales redondeamos su resultado mediante el operador $round.

```
{$project:
    {
        Almacén: "$_id.Almacén",
        Mes: "$_id.Mes",
        _id: 0,
        gananciasNeto: "$gananciasTotales",
        costesTotales: "$costesTotales",
        beneficioNeto: {$round: {$subtract: ["$gananciasTotales","$costesTotales"]}},
        impuestos: {$round: {$multiply: [{$subtract: ["$gananciasTotales","$costesTotales"]},0.21]}},
        beneficioTotal: {$round: {$multiply: [{$subtract: ["$gananciasTotales","$costesTotales"]},1.21]}}
    }
},
```

4) Sort

Ordena el documento en dos fases, en una primera fase ordena según el almacén alfabéticamente y en una segunda fase ordena según el mes de menor a mayor.

```
{$sort:
    {"Almacén":1, "Mes":1}
},
```

5) Segundo Match

Este segundo match selecciona a nuestros Almacenes a lo largo de “x” meses dependiendo de la rentabilidad que hayan tenido para cada uno de esos meses.

Es decir, saldrán los meses que sean rentables según la función de cada almacén.

Nuestra función utiliza el comando $expr, en este caso pide como criterio que la multiplicación de los costes totales de ese mes de ese almacén por 1.5 sean inferiores al beneficio total obtenido.

```
{$match:
    {$expr: {$lt: [{$multiply: ["$costesTotales",1.5]},"$beneficioTotal"]}}
}
```