

Trabajo Práctico: Agregaciones en MongoDB

1. Preparación de la base de datos

Crear tres colecciones con datos de ejemplo:

clientes

```
db.clientes.insertMany([
  { _id: 1, nombre: "Ana", email: "ana@mail.com" },
  { _id: 2, nombre: "Luis", email: "luis@mail.com" },
  { _id: 3, nombre: "María", email: "maria@mail.com" }
])
```

productos

```
db.productos.insertMany([
  { _id: 101, nombre: "Zapatillas", categoria: "Calzado", precio: 100 },
  { _id: 102, nombre: "Camiseta", categoria: "Ropa", precio: 40 },
  { _id: 103, nombre: "Gorra", categoria: "Accesorios", precio: 20 },
  { _id: 104, nombre: "Pantalón", categoria: "Ropa", precio: 60 }
])
```

ventas

```
db.ventas.insertMany([
  {
    _id: 1,
    clienteId: 1,
    items: [
      { productoId: 101, cantidad: 2 },
      { productoId: 102, cantidad: 1 }
    ]
  },
  {
    _id: 2,
    clienteId: 2,
    items: [
      { productoId: 103, cantidad: 1 },
      { productoId: 104, cantidad: 1 }
    ]
  }
])
```

```
{  
  _id: 2,  
  clienteId: 2,  
  items: [  
    { productoId: 103, cantidad: 3 },  
    { productoId: 104, cantidad: 1 }  
  ]  
},  
{  
  _id: 3,  
  clienteId: 1,  
  items: [  
    { productoId: 104, cantidad: 2 }  
  ]  
},  
{  
  _id: 4,  
  clienteId: 3,  
  items: [  
    { productoId: 101, cantidad: 1 },  
    { productoId: 103, cantidad: 2 }  
  ]  
}  
])
```

2. Parte Teórica

Preguntas

1. ¿Qué diferencia hay entre `find()` y `aggregate()` en MongoDB?

`find()`: consultas simples (filtros, proyecciones, sort, limit). Devuelve documentos tal cual están almacenados.

`aggregate()`: pipeline de etapas para transformar datos (agrupar, unir, desanidar, calcular campos, etc.). Devuelve resultados derivados, no necesariamente documentos originales.

2. ¿Para qué sirve el operador `$unwind` en un pipeline de agregación?

Descompone un array en múltiples documentos, uno por elemento del array. Útil para filtrar, agrupar o contar elementos del array.

3. ¿Cuál es la función de `$lookup` y en qué se parece a un *JOIN* de SQL?

`$lookup` realiza un join entre colecciones. Es similar a un LEFT OUTER JOIN en SQL: mantiene todos los documentos de la colección origen y añade los coincidentes de la colección “foránea”.

4. ¿Qué limitación tiene `$group` respecto a los campos que podemos mostrar?

En `$group` solo se puede emitir:

- el campo `_id` del grupo
- campos calculados con acumuladores (`$sum`, `$avg`, `$first`, `$last`, `$push`, etc.).

No se puede “pasar” campos arbitrarios tal cual; si querés conservar un valor no agregado, usá un acumulador apropiado (p. ej. `$first/$last`) o reinyectalo luego con `$lookup/$set`.

5. ¿Qué diferencia hay entre usar `$sort` antes o después de un `$group`?

Antes de `$group`:

- Cambia el resultado de acumuladores sensibles al orden como `$first` y `$last`.
- Puede ayudar performance si el sort usa índice y combinás con `$match/$limit` (reduce datos que llegan al group).
- Puede ser costoso si ordenás un set grande sin índice (memoria/tiempo).

Después de `$group`:

- Ordenás el resultado agregado final (p. ej., ordenar por total por cliente).
- Suele ser más barato porque ya trabajás sobre menos filas (los grupos).



3. Parte Práctica

Ejercicio 1 – Listar todas las ventas con detalle

Armar un pipeline que muestre, para cada ítem vendido:

- Nombre del cliente
- Producto
- Categoría
- Cantidad
- Precio unitario
- Total del ítem (`cantidad * precio`)

The screenshot shows a MongoDB aggregation pipeline interface. The left pane displays the pipeline stages in JSON format, and the right pane shows the resulting sample documents.

```

1  [
2    { $unwind: "$items" },
3
4    {
5      // cliente
6      $lookup: {
7        from: "clientes",
8        localField: "clienteId",
9        foreignField: "_id",
10       as: "cliente"
11     }
12   },
13   { $unwind: "$cliente" },
14
15   {
16     // producto
17     $lookup: {
18       from: "productos",
19       localField: "items.productId",
20       foreignField: "_id",
21       as: "producto"
22     }
23   },
24   { $unwind: "$producto" },
25
26   {
27     // salida final
28     $project: {
29       _id: 0,
30       cliente: "$cliente.nombre",
31       producto: "$producto.nombre",
32       categoria: "$producto.categoría",
33       cantidad: "$items.cantidad",
34       precioUnitario: "$producto.precio",
35       totalItem: [
36         {
37           $multiply: [
38             "$items.cantidad",
39             "$producto.precio"
40           ]
41         }
42       ]
43     }
44   }

```

Pipeline Output
Sample of 7 documents

cliente	producto	categoría	cantidad	precioUnitario	totalItem
"Luis"	"Gorra"	"Accesorios"	3	20	60
"Luis"	"Pantalón"	"Ropa"	1	60	60
"Ana"	"Pantalón"	"Ropa"	2	60	120
"Ana"	"Zapatillas"	"Calzado"	1	100	100
"Maria"	"Gorra"	"Accesorios"	2	20	40
"Maria"	"Pantalón"	"Ropa"	1	60	60
"Maria"	"Zapatillas"	"Calzado"	1	100	100

Ejercicio 2 – Gasto total por cliente

Calcular cuánto gastó cada cliente en total (sumando todos sus ítems de todas las ventas).

Lopez Celina COM3

The screenshot shows the MongoDB aggregation pipeline interface. The top navigation bar includes tabs like Sunwind, \$lookup, \$unwind, \$project, \$group, \$lookup, \$unwind, \$project, and buttons for Generate aggregation, Explain, Export, Run, and Options. Below the navigation is a toolbar with Save, Create New, and Export to Language buttons. The main area contains the aggregation pipeline code and its output.

Pipeline Output: Sample of 3 documents

```
totalGastado : 360
cliente : "Ana"

totalGastado : 120
cliente : "Luis"

totalGastado : 140
cliente : "Maria"
```

```
3   { '$unwind': '$items' },
4
5   // Une con productos para obtener el precio unitario
6   {
7     '$lookup': {
8       'from': 'Productos',
9       'localField': 'items.productoId',
10      'foreignField': '_id',
11      'as': 'producto'
12    }
13  },
14  { '$unwind': '$producto' },
15
16  // Calcula el subtotal por item (cantidad * precio)
17  {
18    '$project': {
19      'clienteId': 1,
20      'subtotal': { '$multiply': ['$items.cantidad', '$producto.precio'] }
21    }
22  },
23
24  // Agrupa por cliente sumando todos sus subtotalos
25  {
26    '$group': {
27      '_id': '$clienteId',
28      'totalGastado': { '$sum: '$subtotal' }
29    }
30  },
31
32  // Une con clientes para mostrar el nombre
33  {
34    '$lookup': {
35      'from': 'Clientes',
36      'localField': '_id',
37      'foreignField': '_id',
38      'as': 'cliente'
39    }
40  },
41  { '$unwind': '$cliente' },
42
43  // Proyecta la salida final
44  {
45    '$project': {
```

Ejercicio 3 – Gasto por cliente y categoría

Calcular el gasto total de cada cliente, pero discriminado por categoría de producto.

The screenshot shows the MongoDB aggregation pipeline interface. The top navigation bar includes tabs like Sunwind, \$lookup, \$unwind, \$project, \$group, \$lookup, \$unwind, \$project, and buttons for Generate aggregation, Explain, Export, Run, and Options. Below the navigation is a toolbar with Save, Create New, and Export to Language buttons. The main area contains the aggregation pipeline code and its output.

Pipeline Output: Sample of 6 documents

```
totalGastado : 200
cliente : "Ana"
categoria : "Calzado"

totalGastado : 160
cliente : "Ana"
categoria : "Ropa"

totalGastado : 60
cliente : "Luis"
categoria : "Ropa"

totalGastado : 60
cliente : "Luis"
categoria : "Accesorios"

totalGastado : 100
cliente : "Maria"
categoria : "Calzado"

totalGastado : 40
cliente : "Maria"
categoria : "Accesorios"
```

```
7   'localField': 'items.productoId',
8   'foreignField': '_id',
9   'as': 'producto'
10  },
11  { '$unwind': '$producto' },
12
13  { // subtotal por item
14  '$project': {
15    'clienteId': 1,
16    'categoria': '$producto.categoría',
17    'subtotal': { '$multiply': ['$items.cantidad', '$producto.precio'] }
18  },
19
20  },
21
22  { // agrupa por cliente + categoria
23  '$group': {
24    '_id': { 'clienteId': '$clienteId', 'categoria': '$categoria' },
25    'totalGastado': { '$sum: '$subtotal' }
26  },
27  },
28
29  { // nombre del cliente
30  '$lookup': {
31    'from': 'Clientes',
32    'localField': '_id.clienteId',
33    'foreignField': '_id',
34    'as': 'cliente'
35  },
36  },
37  { '$unwind': '$cliente' },
38
39  { // salida final
40  '$project': {
41    '_id': 0,
42    'cliente': '$cliente.nombre',
43    'categoria': '_id.categoría',
44    'totalGastado': 1
45  },
46  },
47
48  // opcional: ordenar por cliente y total descendente
49  { '$sort': { 'cliente': 1, 'totalGastado': -1 } }
```

Ejercicio 4 – Productos más vendidos

Armar un pipeline que muestre:

- Nombre del producto
- Cantidad total vendida
- Ordenado de mayor a menor cantidad

The screenshot shows a MongoDB aggregation pipeline interface. At the top, there are tabs for \$unwind, \$lookup, \$unwind, \$group, \$sort, and \$project. Below the tabs are buttons for 'SAVE', 'CREATE NEW', 'EXPORT TO LANGUAGE', 'PREVIEW', 'STAGES', and 'TEXT'. The pipeline code is as follows:

```

1  [
2    // 1. Desarma los items
3    { $unwind: "$items" },
4
5    // 2. Une con productos para obtener nombre
6    {
7      $lookup: {
8        from: "productos",
9        localField: "items.productId",
10       foreignField: "_id",
11       as: "producto"
12     }
13   },
14   { $unwind: "$producto" },
15
16   // 3. Agrupa por producto sumando las cantidades
17   {
18     $group: {
19       _id: "$producto._id",
20       producto: { $first: "$producto.nombre" },
21       cantidadVendida: { $sum: "$items.cantidad" }
22     }
23   },
24
25   // 4. Ordena de mayor a menor cantidad
26   { $sort: { cantidadVendida: -1 } },
27
28   // 5. Proyección final
29   {
30     $project: {
31       _id: 0,
32       producto: 1,
33       cantidadVendida: 1
34     }
35   }
36 ]
37

```

The 'PIPELINE OUTPUT' section shows a sample of 4 documents:

- producto : "Gorra" cantidadVendida : 5
- producto : "Pantalón" cantidadVendida : 3
- producto : "Zapatillas" cantidadVendida : 3
- producto : "Camiseta" cantidadVendida : 1

Ejercicio 5 – Clientes que compraron más de \$200

Usando `$match` después de un `$group`, obtener únicamente los clientes cuyo gasto total supere los \$200.

The screenshot shows a MongoDB aggregation pipeline editor interface. At the top, there are buttons for 'SAVE', '+ CREATE NEW', and 'EXPORT TO LANGUAGE'. The main area contains the following MongoDB aggregation pipeline code:

```

3   { $unwind: "$items" },
4
5   // 2. Une con productos para obtener precio
6   {
7     $lookup: {
8       from: "productos",
9       localField: "items.productId",
10      foreignField: "_id",
11      as: "producto"
12    }
13  },
14  { $unwind: "$producto" },
15
16  // 3. Calcula subtotal por ítem
17  {
18    $project: {
19      clienteId: 1,
20      subtotal: { $multiply: ["$items.cantidad", "$producto.precio"] }
21    }
22  },
23
24  // 4. Agrupa por cliente
25  {
26    $group: {
27      _id: "$clienteId",
28      totalGastado: { $sum: "$subtotal" }
29    }
30  },
31
32  // 5. Filtra los que gastaron más de $200
33  {
34    $match: {
35      totalGastado: { $gt: 200 }
36    }
37  },
38
39  // 6. Une con clientes para mostrar nombre
40  {
41    $lookup: {
42      from: "clientes",
43      localField: "_id",
44      foreignField: "_id",
45      as: "cliente"
46    }
47  }

```

On the right side, there is a 'PIPELINE OUTPUT' section titled 'Sample of 1 document' containing the following JSON data:

```

totalGastado : 360
cliente : "Ana"

```

Ejercicio 6 – Top 1 producto más vendido por categoría

Para cada categoría, mostrar el producto más vendido en cantidad.

Lopez Celina COM3

top1 ventas pr...

```
2   { $unwind: "$items" },
3
4   {
5     $lookup: {
6       from: "productos",
7       localField: "items.productId",
8       foreignField: "_id",
9       as: "producto"
10    }
11  },
12  { $unwind: "$producto" },
13
14 // Cantidad vendida por producto dentro de su categoría
15  {
16    $group: {
17      _id: {
18        categoria: "$producto.categoría",
19        productId: "$producto._id",
20        nombre: "$producto.nombre"
21      },
22      cantidadVendida: { $sum: "$items.cantidad" }
23    }
24  },
25
26 // Ranking por categoría (mayor cantidad primero)
27  {
28    $setWindowFields: {
29      partitionBy: "$_id.categoría",
30      sortBy: { cantidadVendida: -1 },
31      output: { rank: { $rank: {} } }
32    }
33  },
34
35 // Top 1 por categoría (incluye empates en 1)
36  { $match: { rank: 1 } },
37
38 // Salida limpia
39  {
40    $project: {
41      _id: 0,
42      categoria: "$_id.categoría",
43      producto: "$_id.nombre",
44      cantidadVendida: 1
45  }
```

Pipeline Output
Sample of 3 documents

```
cantidadVendida : 5
categoría : "Accesorios"
producto : "Gorra"

cantidadVendida : 3
categoría : "Calzado"
producto : "Zapatillas"

cantidadVendida : 3
categoría : "Ropa"
producto : "Pantalón"
```

¿Cómo harías para calcular el **ticket promedio** (promedio de gasto por venta)?

```
1  [
2   { $unwind: "$items" },
3
4   {
5     $lookup: {
6       from: "productos",
7       localField: "items.productId",
8       foreignField: "_id",
9       as: "prod"
10    }
11  },
12  { $unwind: "$prod" },
13
14 // Subtotal por ítem dentro de la venta
15  {
16    $project: {
17      totalItem: { $multiply: ["$items.cantidad", "$prod.precio"] }
18    }
19  },
20
21 // Total por venta (_id de ventas)
22  {
23    $group: {
24      _id: "$_id",
25      totalVenta: { $sum: "$totalItem" }
26    }
27  },
28
29 // Ticket promedio (promedio de totalVenta)
30  {
31    $group: {
32      _id: null,
33      ticketPromedio: { $avg: "$totalVenta" },
34      ventasConsideradas: { $sum: 1 }
35    }
36  },
37
38 { $project: { _id: 0, ticketPromedio: 1, ventasConsideradas: 1 } }
```

Pipeline Output
Sample of 1 document

```
ticketPromedio : 155
ventasConsideradas : 4
```

¿Qué diferencia hay entre hacer `$lookup` con un solo campo y hacer `$lookup` sobre un array con `$unwind`?

La diferencia es principalmente el nivel de detalle que se puede obtener.

`$lookup` con campo único → relación directa 1 a 1.

`$lookup` sobre array → devuelve todos los matches juntos, pero no ítem por ítem.

`$unwind + $lookup` → relación 1 a 1 por cada elemento del array, ideal para cálculos detallados.