# TP 2 Aggregation Framework en MongoDB

1. Realizar una consulta que devuelva la siguiente información: Región y cantidad total de productos vendidos a clientes de esa Región.

```
> db.facturas.aggregate([
  { $unwind: "$item" },
    { $group: {
        _id: "$cliente.region",
        totalProductos: { $sum: { $multiply: ["$item.cantidad", 1] } }
      }
    }
  ])
< {
    _id: 'CABA',
    totalProductos: 282
  }
  {
    _id: 'CENTRO',
    totalProductos: 180
  }
  {
    _id: 'NOA',
    totalProductos: 14
  }
  {
    _id: 'NEA',
    totalProductos: 420
  }
```

2. Basado en la consulta del punto 1, mostrar sólo la región que tenga el menor ingreso.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.region",
        totalIngreso: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } }
      }
    },
    { $sort: { totalIngreso: 1 } },
    { $limit: 1 }
  ])
< {
    _id: 'NOA',
    totalIngreso: 9800
  }
```

3. Basado en la consulta del punto 1, mostrar sólo las regiones que tengan una cantidad de productos vendidos superior a 300.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.region",
        totalProductos: { $sum: "$item.cantidad" }
      }
    },
    { $match: { totalProductos: { $gt: 300 } } }
  ])
< {
    _id: 'NEA',
    totalProductos: 420
  }
```

4. Se requiere obtener un reporte que contenga la siguiente información, nro. cuit, apellido y nombre y región y cantidad de facturas, ordenado por apellido.

```
> db.facturas.aggregate([
    { $group: {
        _id: { cuit: "$cliente.cuit", apellido: "$cliente.apellido", nombre: "$cliente.nombre", region: "$cliente.region" },
        cantidadFacturas: { $sum: 1 }
      }
    },
    { $sort: { "_id.apellido": 1 } }
  ])
```

```
< {
    _id: {
      cuit: 2729887543,
      apellido: 'Lavagno',
      nombre: 'Soledad',
      region: 'NOA'
    },
    cantidadFacturas: 14
  }
  {
    _id: {
      cuit: 2740488484,
      apellido: 'Malinez',
      nombre: 'Marina',
      region: 'CENTRO'
    },
    cantidadFacturas: 15
  }
  {
    _id: {
      cuit: 2029889382,
      apellido: 'Manoni',
      nombre: 'Juan Manuel',
      region: 'NEA'
    },
    cantidadFacturas: 42
  }
  {
    _id: {
      cuit: 2038373771,
      apellido: 'Zavasi',
      nombre: 'Martin',
      region: 'CABA'
    },
    cantidadFacturas: 29
  }
```

Duplicac

5. Basados en la consulta del punto 4 informar sólo los clientes con número de CUIT mayor a 2700000000.

```
> db.facturas.aggregate([
    { $group: {
        _id: { cuit: "$cliente.cuit", apellido: "$cliente.apellido", nombre: "$cliente.nombre", region: "$cliente.region" },
        cantidadFacturas: { $sum: 1 }
      }
    },
    { $match: { "_id.cuit": { $gt: 2700000000 } } }
  ])
< {
    _id: {
      cuit: 2740488484,
      apellido: 'Malinez',
      nombre: 'Marina',
      region: 'CENTRO'
    },
    cantidadFacturas: 15
  }
  {
    _id: {
      cuit: 2729887543,
      apellido: 'Lavagno',
      nombre: 'Soledad',
      region: 'NOA'
    },
    cantidadFacturas: 14
  }
```

6. Basados en la consulta del punto 5 informar solamente la cantidad de clientes que cumplen con esta condición.

```
> db.facturas.aggregate([
    { $group: {
        _id: "$cliente.cuit"
      }
    },
    { $match: { "_id": { $gt: 2700000000 } } },
    { $count: "cantidadClientes" }
  ])
< {
    cantidadClientes: 2
  }
```

7. Se requiere realizar una consulta que devuelva la siguiente información: producto y cantidad de facturas en las que lo compraron, ordenado por cantidad de facturas

descendente.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$item.producto",
        cantidadFacturas: { $sum: 1 }
      }
    },
    { $sort: { cantidadFacturas: -1 } }
  ])
< {
    _id: 'TALADRO 12mm',
    cantidadFacturas: 43
  }
  {
    _id: 'CORREA 10mm',
    cantidadFacturas: 29
  }
  {
    _id: 'TUERCA 2mm',
    cantidadFacturas: 28
  }
  {
    _id: 'TUERCA 5mm',
    cantidadFacturas: 28
  }
  {
    _id: 'SET HERRAMIENTAS',
    cantidadFacturas: 28
  }
  {
    _id: ' CORREA 12mm',
    cantidadFacturas: 15
  }
```

8. Obtener la cantidad total comprada así como también los ingresos totales para cada producto.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$item.producto",
        cantidadTotal: { $sum: "$item.cantidad" },
        ingresosTotales: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } }
      }
    }
  ])
```

```
< {
    _id: 'TALADRO 12mm',
    cantidadTotal: 43,
    ingresosTotales: 21070
  }
  {
    _id: 'TUERCA 2mm',
    cantidadTotal: 112,
    ingresosTotales: 6720
  }
  {
    _id: 'TUERCA 5mm',
    cantidadTotal: 350,
    ingresosTotales: 31500
  }
  {
    _id: 'SET HERRAMIENTAS',
    cantidadTotal: 28,
    ingresosTotales: 19600
  }
  {
    _id: ' CORREA 12mm',
    cantidadTotal: 165,
    ingresosTotales: 2970
  }
  {
    _id: 'CORREA 10mm',
    cantidadTotal: 198,
    ingresosTotales: 26532
  }
```

9. Idem el punto anterior, ordenar por ingresos en forma ascendente, saltear el 1ro y mostrar 2do y 3ro.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$item.producto",
        cantidadTotal: { $sum: "$item.cantidad" },
        ingresosTotales: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } }
      }
    },
    { $sort: { ingresosTotales: 1 } },
    { $skip: 1 },
    { $limit: 2 }
  ])
< {
    _id: 'TUERCA 2mm',
    cantidadTotal: 112,
    ingresosTotales: 6720
  }
  {
    _id: 'SET HERRAMIENTAS',
    cantidadTotal: 28,
    ingresosTotales: 19600
  }
```

10. Obtener todos productos junto con un array de las personas que lo compraron. En este array deberá haber solo strings con el nombre completo de la persona. Los documentos entregados como resultado deberán tener la siguiente forma: {producto: “<nombre>”, personas:[“…”, …]}

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$item.producto",
        personas: { $addToSet: { $concat: ["$cliente.nombre", " ", "$cliente.apellido"] } }
      }
    },
    { $project: {
        producto: "$_id",
        personas: 1,
        _id: 0
      }
    }
  ])
```

```
< {
    personas: [
      'Marina Malinez'
    ],
    producto: ' CORREA 12mm'
  }
  {
    personas: [
      'Martin Zavasi'
    ],
    producto: 'CORREA 10mm'
  }
  {
    personas: [
      'Juan Manuel Manoni'
    ],
    producto: 'TUERCA 5mm'
  }
  {
    personas: [
      'Juan Manuel Manoni',
      'Martin Zavasi'
    ],
    producto: 'TUERCA 2mm'
  }
  {
    personas: [
      'Juan Manuel Manoni',
      'Marina Malinez'
    ],
    producto: 'TALADRO 12mm'
  }
  {
    personas: [
      'Soledad Lavagno',
```

```
{
  personas: [
    'Soledad Lavagno',
    'Juan Manuel Manoni'
  ],
  producto: 'SET HERRAMIENTAS'
}
```

11. Obtener los productos ordenados en forma descendente por la cantidad de diferentes personas que los compraron.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$item.producto",
        clientesUnicos: { $addToSet: "$cliente.cuit" }
      }
    },
    { $project: {
        _id: 1,
        cantidadPersonas: { $size: "$clientesUnicos" }
      }
    },
    { $sort: { cantidadPersonas: -1 } }
  ])
```

```
< {
    _id: 'TUERCA 2mm',
    cantidadPersonas: 2
  }
  {
    _id: 'TALADRO 12mm',
    cantidadPersonas: 2
  }
  {
    _id: 'SET HERRAMIENTAS',
    cantidadPersonas: 2
  }
  {
    _id: ' CORREA 12mm',
    cantidadPersonas: 1
  }
  {
    _id: 'CORREA 10mm',
    cantidadPersonas: 1
  }
  {
    _id: 'TUERCA 5mm',
    cantidadPersonas: 1
  }
```

12. Obtener el total gastado por persona y mostrar solo los que gastaron más de 3100000. Los documentos devueltos deben tener el nombre completo del cliente y el total gastado:
{cliente:”<nombreCompleto>”,total:<num>}

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.cuit",
        total: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } }
      }
    },
    { $match: { total: { $gt: 3100 } } },
    { $project: {
        cliente: { $concat: ["$cliente.nombre", " ", "$cliente.apellido"] },
        total: 1
      }
    }
  ])
< {
    _id: 2740488484,
    total: 10320,
    cliente: null
  }
  {
    _id: 2038373771,
    total: 31572,
    cliente: null
  }
  {
    _id: 2729887543,
    total: 9800,
    cliente: null
  }
  {
    _id: 2029889382,
    total: 56700,
    cliente: null
  }
```

13. Obtener el promedio de gasto por factura por cada región.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.region",
        totalFacturado: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } },
        cantidadFacturas: { $sum: 1 }
      }
    },
    { $project: {
        region: "$_id",
        promedioGasto: { $divide: ["$totalFacturado", "$cantidadFacturas"] },
        _id: 0
      }
    }
  ])
< {
    region: 'NEA',
    promedioGasto: 675
  }
  {
    region: 'CABA',
    promedioGasto: 734.2325581395348
  }
  {
    region: 'CENTRO',
    promedioGasto: 344
  }
  {
    region: 'NOA',
    promedioGasto: 700
  }
```

14. Obtener la factura en la que se haya gastado más. En caso de que sean varias obtener la que tenga el número de factura menor.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$nroFactura",
        totalFactura: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } }
      }
    },
    { $sort: { totalFactura: -1, _id: 1 } },
    { $limit: 1 }
  ])
< {
    _id: 1002,
    totalFactura: 1968
  }
```

15. Obtener a los clientes indicando cuánto fue lo que más gastó en una única factura.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.cuit",
        maxGastoFactura: { $max: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } } },
        cliente: { $first: { $concat: ["$cliente.nombre", " ", "$cliente.apellido"] } }
      }
    },
    { $project: {
        cliente: 1,
        maxGastoFactura: 1,
        _id: 0
      }
    }
  ])
< {
    maxGastoFactura: 490,
    cliente: 'Marina Malinez'
  }
  {
    maxGastoFactura: 1608,
    cliente: 'Martin Zavasi'
  }
  {
    maxGastoFactura: 1350,
    cliente: 'Juan Manuel Manoni'
  }
  {
    maxGastoFactura: 700,
    cliente: 'Soledad Lavagno'
  }
```

16. Utilizando MapReduce, indicar la cantidad total comprada de cada ítem. Comparar el resultado con el ejercicio 8.
17. Obtener la información de los clientes que hayan gastado 1960 en una orden junto con el número de orden.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$nroFactura",
        totalFactura: { $sum: { $multiply: ["$item.cantidad", "$item.precio"] } },
        cliente: { $first: { $concat: ["$cliente.nombre", " ", "$cliente.apellido"] } }
      }
    },
    { $match: { totalFactura: { $gte: 1960 } } }
  ])
```

```
< {
    _id: 1072,
    totalFactura: 1968,
    cliente: 'Martin Zavasi'
  }
  {
    _id: 1009,
    totalFactura: 1968,
    cliente: 'Martin Zavasi'
  }
  {
    _id: 1079,
    totalFactura: 1968,
    cliente: 'Martin Zavasi'
  }
  {
    _id: 1010,
    totalFactura: 1960,
    cliente: 'Juan Manuel Manoni'
  }
  {
    _id: 1044,
    totalFactura: 1968,
    cliente: 'Martin Zavasi'
  }
  {
    _id: 1003,
    totalFactura: 1960,
    cliente: 'Juan Manuel Manoni'
  }
  {
    _id: 1037,
    totalFactura: 1968,
    cliente: 'Martin Zavasi'
  }
```

```
{
  _id: 1045,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1017,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1002,
  totalFactura: 1968,
  cliente: 'Martin Zavasi'
}
{
  _id: 1058,
  totalFactura: 1968,
  cliente: 'Martin Zavasi'
}
{
  _id: 1052,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1059,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1073,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
```

```
{
  _id: 1024,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1030,
  totalFactura: 1968,
  cliente: 'Martin Zavasi'
}
{
  _id: 1087,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
{
  _id: 1016,
  totalFactura: 1968,
  cliente: 'Martin Zavasi'
}
{
  _id: 1051,
  totalFactura: 1968,
  cliente: 'Martin Zavasi'
}
{
  _id: 1080,
  totalFactura: 1960,
  cliente: 'Juan Manuel Manoni'
}
Type "it" for more
```

18. En base a la localidad de los clientes, obtener el total facturado por localidad.

```
> db.facturas.aggregate([
    { $unwind: "$item" },
    { $group: {
        _id: "$cliente.region",
        totalFacturado: {
          $sum: {
            $multiply: [
              { $ifNull: ["$item.cantidad", 0] },
              { $ifNull: ["$item.precio", 0] }
            ]
          }
        }
      }
    }
  ])
< {
    _id: 'NEA',
    totalFacturado: 56700
  }
  {
    _id: 'CENTRO',
    totalFacturado: 10320
  }
  {
    _id: 'NOA',
    totalFacturado: 9800
  }
  {
    _id: 'CABA',
    totalFacturado: 31572
  }
```