Lab: MongoDB Aggregation and MapReduce

1. Go the test database using the following command

```
use test
```

2. Use insertMany() to create a collection orders and populate the collection

```
db.orders.insertMany( [
  { cust_id: "A123", amount: 500, status: "A" },
  { cust_id: "A123", amount: 250, status: "A" },
  { cust_id: "B212", amount: 200, status: "A" },
  { cust_id: "A123", amount: 300, status: "D" }
] );
```

3. $match stage

```
db.orders.aggregate( [
    { $match : { status : "A" } }
] );

db.orders.aggregate( [
    { $match : { cust_id: "A123" , status : "A" } }
] );

db.orders.aggregate ( [
    { $match : { $or: [ {amount: {$gte: 300}} , {status : "D"} ] } }
] );
```

   What are the equivalent commands without using $match?

4. $group stage

```
db.orders.aggregate ( [
    { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
] );
```

   What is the equivalent SQL?

5. Aggregation pipeline

```
db.orders.aggregate ( [
    { $match : { status : "A" } },
    { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
] );
```

6. $count

```
db.orders.aggregate ( [
    { $match : { status : "A" } }, { $group: { _id: null, count: { $sum: 1 } } }
] );
```

7. $sort

```
db.orders.aggregate ( [
    { $match : { status : "A" } },
    { $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
    { $sort : { _id : 1 } }
] );
```

8. $min, $max, $avg

```
db.orders.aggregate ( [
    { $match : { status : "A" } },
    { $group: { _id: null, avg_amount: { $avg: "$amount" } } }
] );
```

```
db.orders.aggregate ( [
    { $match : { status : "A" } },
    { $group: {_id: "$cust_id", avg_amount: { $avg: "$amount" } } }
] );
```

What is the difference between the above two commands?

```
db.orders.aggregate ( [
    { $group: { _id: "$cust_id", max_amount: { $max: "$amount" } } }
] );
```

9. $unwind Operator

   a. Delete all documents from inventory
   ```
   db.inventory.deleteMany( {} )
   ```

   b. Insert a document to inventory
   ```
   db.inventory.insertOne( { "_id" : 1, "item" : "ABC1", sizes: [ "S", "M", "L"] } )
   ```

The following aggregation uses the $unwind stage to output a document for each element in the sizes array:

```
db.inventory.aggregate( [ { $unwind : "$sizes" } ] )
```

What is the output?

c. Create a sample collection named inventory2 with the following documents:

```
db.inventory2.insertMany( [
  { "_id" : 1, "item" : "ABC", price: NumberDecimal("80"), "sizes": [ "S", "M", "L" ] },
  { "_id" : 2, "item" : "EFG", price: NumberDecimal("120"), "sizes" : [ ] },
  { "_id" : 3, "item" : "IJK", price: NumberDecimal("160"), "sizes": "M" },
  { "_id" : 4, "item" : "LMN" , price: NumberDecimal("10") },
  { "_id" : 5, "item" : "XYZ", price: NumberDecimal("5.75"), "sizes" : null }
] )
```

What is the output of the following command?

```
db.inventory2.aggregate( [ { $unwind: "$sizes" } ] )
```

What is the output of the following command?

```
db.inventory2.aggregate( [
   { $unwind: { path: "$sizes", preserveNullAndEmptyArrays: true } }
] )
```

The following $unwind operation uses the includeArrayIndex option to include the array index in the output.

```
db.inventory2.aggregate( [
  {
    $unwind:
      {
        path: "$sizes",
        includeArrayIndex: "arrayIndex"
      }
   }
] )
```

You may also try to add preserveNullAndEmptyArrays: true to see what will happen.

```
db.inventory2.aggregate( [
 {
   $unwind:
     {
       path: "$sizes",
       preserveNullAndEmptyArrays: true,
       includeArrayIndex: "arrayIndex"
     }
  }
] )
```

What is the difference?

d. Group by Unwound Values

You may check the following website https://docs.mongodb.com/manual/reference/operator/aggregation/unwind/#group-by-unwound-values for detailed explanation.

Enter the following command

```
db.inventory2.aggregate( [
  // First Stage
  {
    $unwind: { path: "$sizes", preserveNullAndEmptyArrays: true }
  },
  // Second Stage
  {
    $group:
      {
        _id: "$sizes",
        averagePrice: { $avg: "$price" }
      }
  },
  // Third Stage
  {
    $sort: { "averagePrice": -1 }
  }
] )
```

What is the output?

e. Unwind Embedded Arrays

Create a sample collection named sales with the following documents:

```
db.sales.insertMany([
 {
   _id: "1",
   "items" : [
    {
     "name" : "pens",
     "tags" : [ "writing", "office", "school", "stationary" ],
     "price" : NumberDecimal("12.00"),
     "quantity" : NumberInt("5")
    },
    {
     "name" : "envelopes",
     "tags" : [ "stationary", "office" ],
     "price" : NumberDecimal("1.95"),
     "quantity" : NumberInt("8")
    }
   ]
 },
 {
   _id: "2",
   "items" : [
    {
     "name" : "laptop",
     "tags" : [ "office", "electronics" ],
     "price" : NumberDecimal("800.00"),
     "quantity" : NumberInt("1")
    },
    {
     "name" : "notepad",
     "tags" : [ "stationary", "school" ],
     "price" : NumberDecimal("14.95"),
     "quantity" : NumberInt("3")
    }
   ]
 }
] )
```

What are the embedded arrays?

The following operation groups the items sold by their tags and calculates the total sales amount per each tag.

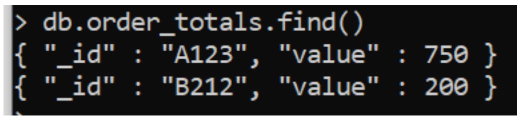
```
db.sales.aggregate( [
  // First Stage
  { $unwind: "$items" },

  // Second Stage
  { $unwind: "$items.tags" },

  // Third Stage
  {
    $group:
      {
        _id: "$items.tags",
        totalSalesAmount:
          {
            $sum: { $multiply: [ "$items.price", "$items.quantity" ] }
          }
      }
  }
] )
```

10. A MapReduce example

```
db.orders.mapReduce(
    function() { emit(this.cust_id , this.amount); },
    function(key, values) { return Array.sum(values) },
    {
            query: { status: "A" },
            out: "order_totals"
    }
);
```

Check the results

```
db.order_totals.find()
```



```
> db.order_totals.find()
{ "_id" : "A123", "value" : 750 }
{ "_id" : "B212", "value" : 200 }
```

Or you can try

```
db.orders.mapReduce(
    function() { emit(this.cust_id , this.amount); },
    function(key,values) { return Array.sum(values) },
    {
            query: { status: "A" },
            out: { inline: 1 }
    }
);
```

11. $lookup for Join

Create a collection orders with the following documents (delete the documents in orders first if there are any):

```
db.orders.insertMany( [
  { _id : 1, item : "almonds", price : 12, quantity : 2 },
  { _id : 2, item : "pecans", price : 20, quantity : 1 },
  { _id : 3  }
] );
```

Create another collection inventory with the following documents (delete the documents in inventory first if there are any):

```
db.inventory.insertMany( [
  { _id : 1, sku : "almonds", description: "product 1", instock : 120 },
  { _id : 2, sku : "bread", description: "product 2", instock : 80 },
  { _id : 3, sku : "cashews", description: "product 3", instock : 60 },
  { _id : 4, sku : "pecans", description: "product 4", instock : 70 },
  { _id : 5, sku : null, description: "Incomplete" },
  { _id : 6 }
] );
```

How to join on item=sku?

```
db.orders.aggregate( [
  {
    $lookup:
      {
        from: "inventory",
        localField: "item",
        foreignField: "sku",
        as: "inventory_docs"
      }
  }
] );
```

The operation would correspond to the following pseudo-SQL statement:

```
SELECT      *, inventory_docs
FROM        orders
WHERE       inventory_docs IN (SELECT *
                               FROM inventory
                               WHERE sku= orders.item);
```

References

https://www.tutorialspoint.com/mongodb/mongodb_aggregation.htm

https://docs.mongodb.com/manual/

https://appdividend.com/2018/10/25/mongodb-aggregate-example-tutorial/

https://appdividend.com/2018/10/26/mongodb-mapreduce-example-tutorial/

https://docs.mongodb.com/manual/reference/method/db.collection.distinct/

https://docs.mongodb.com/manual/reference/operator/aggregation/lookup/