Lab: MongoDB Practice 2

Q1. Start from an empty inventory collection and populate the collection using the following command.

```
db.inventory.insertMany( [
  { item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" },
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "mat", qty: 85, size: { h: 27.9, w: 35.5, uom: "cm" }, status: "A" },
  { item: "mousepad", qty: 75, size: { h: 19, w: 22.85, uom: "cm" }, status: "P" },
  { item: "notebook", qty: 85, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },
  { item: "sketchbook", qty: 80, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "sketch pad", qty: 95, size: { h: 22.85, w: 30.5, uom: "cm" }, status: "A" }
] );
```

1. For the documents with qty greater than 50, obtain the total qty of each status and then sort the results by the status in ascending order.

   ```
   db.inventory.aggregate( [
      { $match: { qty: { $gt: 50 } } },
      { $group: { _id: "$status", total_qty: { $sum: "$qty" } } },
      { $sort: { _id: 1 } }
   ] )
   ```

   The following will write the result of the pipeline to another collection in the same database. $out must be the last stage of an aggregation pipeline

   ```
   db.inventory.aggregate( [
      { $match: { qty: { $gt: 50 } } },
      { $group: { _id: "$status", total_qty: { $sum: "$qty" } } },
      { $sort: { _id: 1 } },
      { $out: "total_qty" }
   ] )
   ```

   The following will write the result of the pipeline to another collection in a different database.

   ```
   db.inventory.aggregate( [
      { $match: { qty: { $gt: 50 } } },
      { $group: { _id: "$status", total_qty: { $sum: "$qty" } } },
      { $sort: { _id: 1 } },
      { $out: { db: "testDB",  coll: "total_qty" } }
   ] )
   ```

2. For the documents with qty greater than 50, first compute the total qty of each status and then find the status with total qty greater than 170.

```
db.inventory.aggregate( [
   { $match: { qty: { $gt: 50 } } },
   { $group: { _id: "$status", total_qty: { $sum: "$qty" } } },
   { $match: {total_qty: { $gt: 170 } } }
] )
```

Q2. Start from an empty inventory collection and populate the collection using the following command.

```
db.inventory.insertMany( [
  { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
  { item: "notebook", instock: [ { warehouse: "C", qty: 55 } ] },
  { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
  { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
  { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
] );
```

1. For the items that are available at two warehouses, find the total stock of each item and sort the results by the total stock in ascending order.

```
db.inventory.aggregate( [
   { $match: { instock: { $size: 2 } } },
   { $unwind: "$instock" },
   { $group: { _id: "$item", total_stock: { $sum: "$instock.qty" } } },
   { $sort: { total_stock: 1 } }
] )
```

We can also use $project as follows.

```
db.inventory.aggregate( [
   { $match: { instock: { $size: 2 } } },
   { $project: { _id: "$item", total_stock: { $sum: "$instock.qty" } } },
   { $sort: { total_stock: 1 } }
] )
```

2. Find the total inventory (the total stock of all items) at each warehouse.

```
db.inventory.aggregate( [
   { $unwind: "$instock" },
   { $group: { _id: "$instock.warehouse", total_inventory: { $sum: "$instock.qty" } } }
] )
```

3. Based on the previous question, find the maximum total inventory of all warehouses.

```
db.inventory.aggregate( [
   { $unwind: "$instock" },
   { $group: { _id: "$instock.warehouse", total_inventory: { $sum: "$instock.qty" } } },
   { $group: { _id: null, max_inventory: { $max: "$total_inventory" } } }
] )
```

You can further add a $project stage to remove the _id field.

4. Find the warehouse(s) that has (have) the maximum total inventory.

It is easy to just find one warehouse as follows.

```
db.inventory.aggregate( [
   { $unwind: "$instock" },
   { $group: { _id: "$instock.warehouse", total_inventory: { $sum: "$instock.qty" } } },
   { $sort: { total_inventory: -1 } },
   { $limit: 1 }
] )
```

To find all such warehouses, we can use $push as follows.

```
db.inventory.aggregate([
   { $unwind: "$instock" },
   { $group: { _id: "$instock.warehouse", total_inventory: { $sum: "$instock.qty" } } },
   { $group: { _id: "$total_inventory", warehouses: { $push: "$_id" } } },
   { $sort: { _id: -1 } },
   { $limit: 1 }
] )
```

We can improve the output format as follows

```
db.inventory.aggregate([
   { $unwind: "$instock" },
   { $group: { _id: "$instock.warehouse", total_inventory: { $sum: "$instock.qty" } } },
   { $group: { _id: "$total_inventory", warehouses: { $push: "$_id" } } },
   { $sort: { _id: -1 } },
   { $limit: 1 },
   { $unwind: "$warehouses" },
   { $project: { warehouse: "$warehouses", total_inventory: "$_id", _id: 0 } }
] )
```