QM 7093: Enterprise Data Systems: NoSQL with MongoDB

Noah L. Schrick

13 December, 2022

Contents

1 MongoDB **2**

2 Insertions and Queries **2**

2.1 Inserting Data 2

2.2 Queries 4

3 Metadata **7**

1 MongoDB

Due to the flexible nature of the project assignment and the ability to take alternate database approaches, MongoDB was chosen as the database implementation. While both MongoDB and CouchDB are document-based NoSQL databases, they each have differing advantages. My primary research focuses and interests revolve around the High-Performance Computing (HPC) space, and MongoDB sees greater usage in this area. MongoDB has greater scalability and better performance than CouchDB, though it does lack the design priorities of availability that CouchDB offers. MongoDB is also one of the most-widely used databases across all models, ranking at position 5 on `https://db-engines.com/en/ranking?utm_source=xp&utm_medium=blog&utm_campaign=content`.

2 Insertions and Queries

2.1 Inserting Data

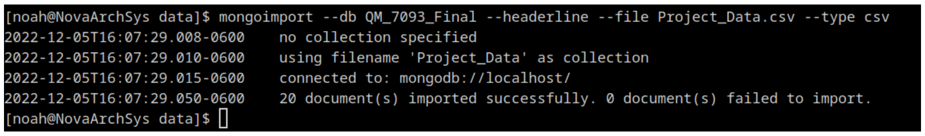
Insert all records from the provided datasheet with the following properties:

- Create all the records with columns StockCode, Description, Quantity, Price, Customer ID and Country (that means you should NOT include invoice and invoice Date in your columns).
- The code for records without customer ID should NOT have a customer ID column.
- Create another column "HighDemand" but ONLY for records with Quantity more than 12 (12 included). In the column put "Yes".

To reduce the amount of manual insertions and minimize the risk of human insertion error, the xlsx datasheet was converted to a csv. Each text cell ("Description", for example) was encapsulated in quotes before the conversion. The delimiter used was a comma (","). Saving the data in a csv format allows for an easy insertion by MongoDB using mongoimport.

```
    mongoimport --db QM_7093_Final --headerline --file Project_Data.csv
--type csv
```

In real-world or large dataset applications, this is not an ideal approach. Inserting data only to subsequently remove it is an inefficient and unoptimized approach. Pre-processing the data to selectively remove unwanted records before insertion would be a better solution.



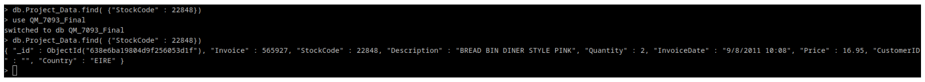
```
[noah@NovaArchSys data]$ mongoimport --db QM_7093_Final --headerline --file Project_Data.csv --type csv
2022-12-05T16:07:29.008-0600    no collection specified
2022-12-05T16:07:29.010-0600    using filename 'Project_Data' as collection
2022-12-05T16:07:29.015-0600    connected to: mongodb://localhost/
2022-12-05T16:07:29.050-0600    20 document(s) imported successfully. 0 document(s) failed to import.
[noah@NovaArchSys data]$ 
```

Figure 1: Part 1.a: Importing from CSV

Removing Invoice and InvoiceDate from the Project_Data collection can be performed with:

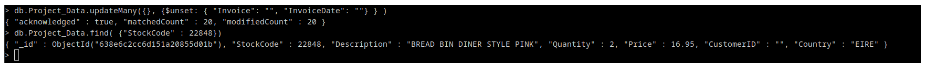
```
db.Project_Data.updateMany({}, {$unset: { "Invoice": "", "InvoiceDate":
""}} )
```

An image of a sample record prior to removing Invoice and Invoice Date can be seen in Figure 2, and an image of the same record after removing the fields can be seen in Figure 3.



```
> db.Project_Data.find( {"StockCode" : 22848})
> use QM_7093_Final
switched to db QM_7093_Final
> db.Project_Data.find( {"StockCode" : 22848})
{ "_id" : ObjectId("638e6ba19804d9f256053d1f"), "Invoice" : 565927, "StockCode" : 22848, "Description" : "BREAD BIN DINER STYLE PINK", "Quantity" : 2, "InvoiceDate" : "9/8/2011 10:08", "Price" : 16.95, "CustomerID
" : "", "Country" : "EIRE" }
> 
```

Figure 2: Collection Sample Prior to Removing Invoice and Invoice Date

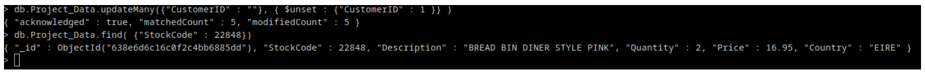


```
> db.Project_Data.updateMany({}, {$unset: { "Invoice": "", "InvoiceDate": ""} } )
{ "acknowledged" : true, "matchedCount" : 20, "modifiedCount" : 20 }
> db.Project_Data.find( {"StockCode" : 22848})
{ "_id" : ObjectId("638e6c2cc6d151a20855d01b"), "StockCode" : 22848, "Description" : "BREAD BIN DINER STYLE PINK", "Quantity" : 2, "Price" : 16.95, "CustomerID" : "", "Country" : "EIRE" }
> 
```

Figure 3: Collection Sample After Removing Invoice and Invoice Date

Removing the CutomerID field when empty can be performed with:

```
db.Project_Data.updateMany({"CustomerID" : ""}, { $unset : {"CustomerID"
: 1 } } )
```



```
> db.Project_Data.updateMany({"CustomerID" : ""}, { $unset : {"CustomerID" : 1 }} )
{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }
> db.Project_Data.find( {"StockCode" : 22848})
{ "_id" : ObjectId("638e6d6c16c0f2c4bb6885dd"), "StockCode" : 22848, "Description" : "BREAD BIN DINER STYLE PINK", "Quantity" : 2, "Price" : 16.95, "Country" : "EIRE" }
> 
```

Figure 4: Collection Sample After Removing Empty CustomerID fields

Adding the HighDemand column can be performed with an aggregate function. If the condition is met, the true value adds "Yes" to the value of the newly added field. $$REMOVE is a built-in mongo indicator to suppress. If the condition is not met, the field is not added to the record. By default, aggregates only read and do not write. Adding $out tells mongo to write the results. In this case, we have told mongo to write back to the Project_Data collection.

```
db.Project_Data.aggregate([
    {
        $addFields: {
            HighDemand: {
              $cond: [
                { $gt: [ "$Quantity", 11 ] },
              "Yes",
              "$$REMOVE"
              ]
            }
        }
    },
    {
        $out: "Project_Data"
    }
    ]).pretty()
```

2.2 Queries

Question 1: How many records have the column "HighDemand"? (Must have a code to answer this, one way to answer this is to have a code that displays all the records except those with the column HighDemand and then subtract the number from total number of records)

```
db.Project_Data.count({"HighDemand": "Yes"})
```

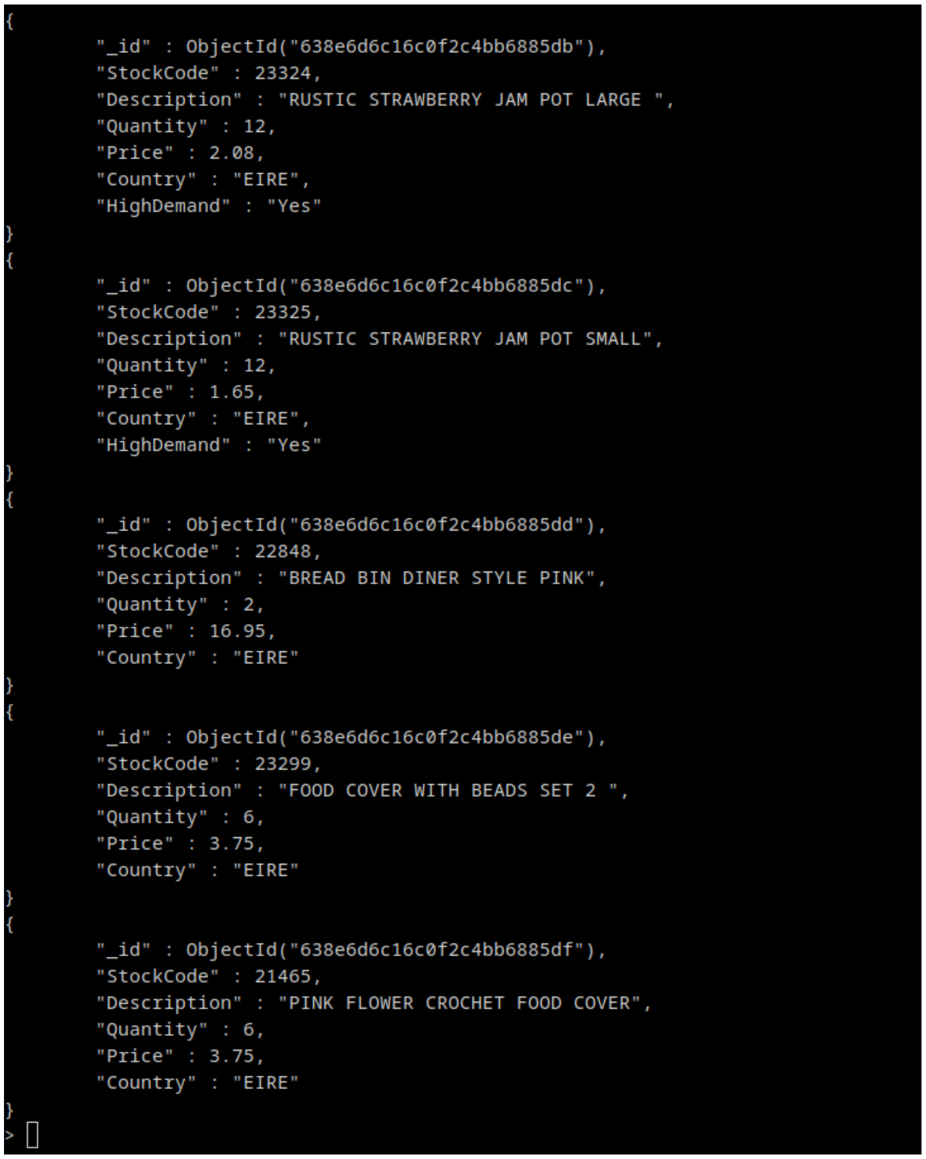
Answer: 8.

A partial image of the records with column HighDemand can be seen in Figure 5.

Question 2: Display the records with price more than 4 (4 excluded)

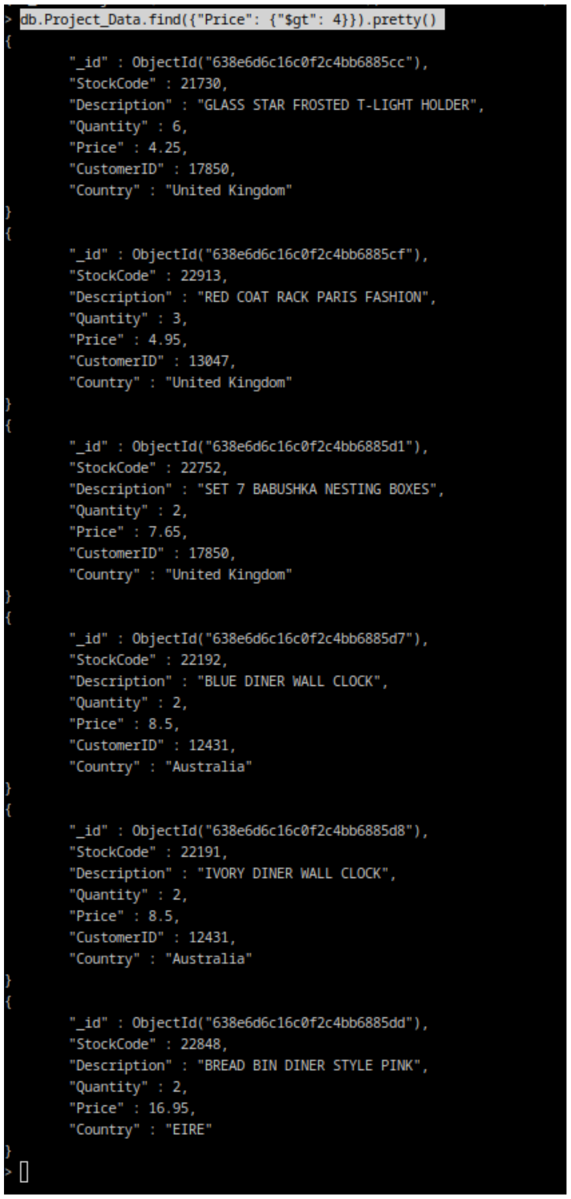
```
db.Project_Data.find({"Price": {"$gt": 4}}).pretty()
```

Total number of records with price greater than four: 6.



```
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885db"),
        "StockCode" : 23324,
        "Description" : "RUSTIC STRAWBERRY JAM POT LARGE ",
        "Quantity" : 12,
        "Price" : 2.08,
        "Country" : "EIRE",
        "HighDemand" : "Yes"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885dc"),
        "StockCode" : 23325,
        "Description" : "RUSTIC STRAWBERRY JAM POT SMALL",
        "Quantity" : 12,
        "Price" : 1.65,
        "Country" : "EIRE",
        "HighDemand" : "Yes"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885dd"),
        "StockCode" : 22848,
        "Description" : "BREAD BIN DINER STYLE PINK",
        "Quantity" : 2,
        "Price" : 16.95,
        "Country" : "EIRE"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885de"),
        "StockCode" : 23299,
        "Description" : "FOOD COVER WITH BEADS SET 2 ",
        "Quantity" : 6,
        "Price" : 3.75,
        "Country" : "EIRE"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885df"),
        "StockCode" : 21465,
        "Description" : "PINK FLOWER CROCHET FOOD COVER",
        "Quantity" : 6,
        "Price" : 3.75,
        "Country" : "EIRE"
}
> 
```

Figure 5: Records with HighDemand



```
> db.Project_Data.find({"Price": {"$gt": 4}}).pretty()
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885cc"),
        "StockCode" : 21730,
        "Description" : "GLASS STAR FROSTED T-LIGHT HOLDER",
        "Quantity" : 6,
        "Price" : 4.25,
        "CustomerID" : 17850,
        "Country" : "United Kingdom"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885cf"),
        "StockCode" : 22913,
        "Description" : "RED COAT RACK PARIS FASHION",
        "Quantity" : 3,
        "Price" : 4.95,
        "CustomerID" : 13047,
        "Country" : "United Kingdom"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885d1"),
        "StockCode" : 22752,
        "Description" : "SET 7 BABUSHKA NESTING BOXES",
        "Quantity" : 2,
        "Price" : 7.65,
        "CustomerID" : 17850,
        "Country" : "United Kingdom"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885d7"),
        "StockCode" : 22192,
        "Description" : "BLUE DINER WALL CLOCK",
        "Quantity" : 2,
        "Price" : 8.5,
        "CustomerID" : 12431,
        "Country" : "Australia"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885d8"),
        "StockCode" : 22191,
        "Description" : "IVORY DINER WALL CLOCK",
        "Quantity" : 2,
        "Price" : 8.5,
        "CustomerID" : 12431,
        "Country" : "Australia"
}
{
        "_id" : ObjectId("638e6d6c16c0f2c4bb6885dd"),
        "StockCode" : 22848,
        "Description" : "BREAD BIN DINER STYLE PINK",
        "Quantity" : 2,
        "Price" : 16.95,
        "Country" : "EIRE"
}
> 
```

Figure 6: Records with Price Greater than Four

3 Metadata

```
{
  "metadata": [
    {"key": "InvoiceReceipt", "value": "IN-C123456.pdf"},
    {"key": "File size", "value": 32764},
    {"key": "MIME type", "value": "application/pdf"},
    {"key": "CancellationStatus", "value": "true"},
    {"key": "Author", "value": {"LName": "Schrick", "FName": "Noah"}},
    {"key": "Security", "value": "false"},
    {"key": "Fonts", "value": "Calibri"},
    {"key": "URL", "value": ""},
    {"key": "RevisionTimestamp", "value": ISODate("2022-12-08T10:01:00Z")},
    {"key": "ItemCategory", "value": "furniture"},
    {"key": "ItemWeight", "value": "20"},
    {"key": "CustomerStanding", "value": "good"},
    {"key": "PaymentMethod", "value": "Bank"},
    {"key": "CreditCardVendor", "value": ""},
    {"key": "Origin", "value": "USA"},
    {"key": "Expedited", "value": "false"},
    {"key": "HoldStatus", "value": ""},
    {"key": "Wholesaler", "value": "true"}
  ]
}
```