Benjamin Katz

1)How many documents in the collection? You must use aggregation to implement this query.

Single purpose aggregation method

```
test> db.computer.count()
DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
200
```

2)How many documents are associated with each brand? The resultset must be sorted in descending order of “number of documents”.

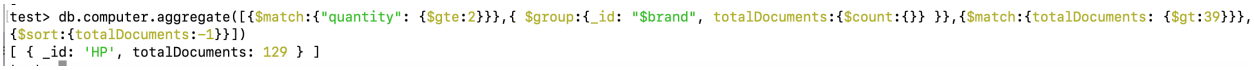
```
test> db.computer.aggregate([{ $group:{_id: "$brand", totalDocuments:{$count:{}} }},{$sort:{totalDocuments:-1}}])
[
  { _id: 'HP', totalDocuments: 147 },
  { _id: 'Lenovo', totalDocuments: 42 },
  { _id: 'Apple', totalDocuments: 6 },
  { _id: 'Asus', totalDocuments: 3 },
  { _id: 'Dell', totalDocuments: 2 }
]
```

3)How many documents are associated with each brand, but include only documents that have at least two in stock for that type? The result-set must be sorted in descending order of “number of documents”.

```
test> db.computer.aggregate([{$match:{"quantity": {$gte:2}}},{ $group:{_id: "$brand", totalDocuments:{$count:{}} }},{$sort:{totalDocuments:-1}}])
[
  { _id: 'HP', totalDocuments: 129 },
  { _id: 'Lenovo', totalDocuments: 39 },
  { _id: 'Apple', totalDocuments: 6 },
  { _id: 'Asus', totalDocuments: 2 },
  { _id: 'Dell', totalDocuments: 2 }
]
```

4)How many documents are associated with each brand, include only documents that have at least two in stock for that type, and only if the total number of such documents exceeds thirty-none? The result-set must be sorted in descending order of “number of documents”.

```
test> db.computer.aggregate([{$match:{"quantity": {$gte:2}}},{ $group:{_id: "$brand", totalDocuments:{$count:{}} }},{$match:{totalDocuments: {$gt:39}}},{$sort:{totalDocuments:-1}}])
[
  { _id: 'HP', totalDocuments: 129 }
```



5)For each brand in the data-set, what is the maximum amount of memory for that brand? Each tuple in the result-set must include the brand-name and the amount of memory. The result-set must be sorted in decreasing order of memory, and then by increasing lexicographical order of brand name.

```
test> db.computer.aggregate([
...     {
...     $unwind: "$attributes"
...     },
...     {
...         $match:
...         {
...             "attributes.attribute_name": "memory"
...         }
...     },
...     {
...         $group:
...         {
...             _id: "$brand",
...             maxRam: {
...                 $max: {
...                     $toInt: {$substr: [ "$attributes.attribute_value", 0,
...                     {$subtract:[{$strLenCP:
"$attributes.attribute_value"},2]}]}
...                 }
...             }
...         }
...     },
...     {
...         $sort:{maxRam:-1, _id: 1}
...     }
```

...])
[
  { _id: 'HP', maxRam: 32 },
  { _id: 'Lenovo', maxRam: 32 },
  { _id: 'Apple', maxRam: 16 },
  { _id: 'Asus', maxRam: 16 },
  { _id: 'Dell', maxRam: 16 }

```
test> db.computer.aggregate([
...     {
...     $unwind: "$attributes"
...     },
...     {
...             $match:
...             {
...                     "attributes.attribute_name": "memory"
...             }
...     },
...     {
...             $group:
...             {
...                      _id: "$brand",
...                      maxRam: {
...                             $max: {
...                                     $toInt: {$substr: [ "$attributes.attribute_value", 0,
...                                     {$subtract:[{$strLenCP:  "$attributes.attribute_value"},2]}]}
...                             }
...                     }
...             }
...     },
...     {
...             $sort:{maxRam:-1, _id: 1}
...     }
... ])
[
  { _id: 'HP', maxRam: 32 },
  { _id: 'Lenovo', maxRam: 32 },
  { _id: 'Apple', maxRam: 16 },
  { _id: 'Asus', maxRam: 16 },
  { _id: 'Dell', maxRam: 16 }
]
```