Weight:

| Map | Maps all normalized ingredients and amounts |
| --- | --- |
| var mapFunc = function () { emit(this.recipe\_title, [this.normalized\_ingredients, this.normalized\_amounts]);} |  |
| Reduce | Reduce filters all ingredients to only show ingredients that are measured in weight. For every measurement, it checks if it has a measurement (by length of split amount array, if its 2 then has a number and a measurement) and if it exists in the weight\_measurements array (measured by weight). It pushes those values into amounts array, and pushes their indexes into weight\_indexes array, then uses those indexes to push matching ingredients into ingredients array. Finally, returns amounts and ingredients. |
| var reduceFunc = function (key, values) { var weight\_measurements = ["oz","pound","ounce","lb","kg","gram"]; var amounts = []; var ingredients = []; var weight\_indexes =[]; for(i = 0; i < values[0][1].length; i++){ var currAmountArr = values[0][1][i].split(" "); if (currAmountArr.length == 2) { for(g = 0; g < weight\_measurements.length; g++) {if (weight\_measurements[g] == currAmountArr[1]) {amounts.push(values[0][1][i]); weight\_indexes.push(i)}}}} for(j = 0; j < weight\_indexes.length; j++){ ingredients.push(values[0][0][weight\_indexes[j]]); } return [ingredients, amounts];} |  |
| Finalize and run mapReduce | Finalize returns only ingredients  (This could’ve been done in reduce, but to check the function, was done here) |
| Finalize:  var finalizeFunc = function(key, values) { return values[0]; }  —-----  Run mapReduce:  db.recipes.mapReduce(mapFunc,reduceFunc,{ out:"weight\_map\_reduce\_result",finalize: finalizeFunc}) |  |
| Aggregate to find distinct ingredients |  |
| db.weight\_map\_reduce\_result.distinct("value") |  |

Unit:

| Map | Maps all normalized ingredients and amounts |
| --- | --- |
| var mapFunc = function () { emit(this.recipe\_title, [this.normalized\_ingredients, this.normalized\_amounts]);} |  |
| Reduce | Reduce filters all ingredients to only show ingredients that are measured in units. For every measurement, it checks if it doesn’t have a measurement (by length of split amount array, if its 2 then has a number and a measurement) and if the amount isn’t empty (like in case of ‘pinch of salt’, if it isn’t empty then its a number). It pushes those values into amounts array, and pushes their indexes into weight\_indexes array, then uses those indexes to push matching ingredients into ingredients array. Finally, returns amounts and ingredients. |
| var reduceFunc = function (key, values) { var amounts = []; var ingredients = []; var weight\_indexes =[]; for(i = 0; i < values[0][1].length; i++){ var currAmountArr = values[0][1][i].split(" "); if (currAmountArr.length == 1 && currAmountArr[0] != "") { amounts.push(values[0][1][i]); weight\_indexes.push(i)}} for(j = 0; j < weight\_indexes.length; j++){ ingredients.push(values[0][0][weight\_indexes[j]]); } return [ingredients, amounts];} |  |
| Finalize and run mapReduce | Finalize returns only ingredients  (This could’ve been done in reduce, but to check the function, was done here) |
| Finalize:  var finalizeFunc = function(key, values) { return values[0]; }  —-----  Run mapReduce:  db.recipes.mapReduce(mapFunc,reduceFunc,{ out:"unit\_map\_reduce\_result",finalize: finalizeFunc}) |  |
| Aggregate to find distinct ingredients |  |
| db.unit\_map\_reduce\_result.distinct("value") |  |