Using JavaScript (and NOT aggregation framework), we will explore “Students” csv file.

Script to Import from csv file: mongoimport --db students --collection temp --type csv --file /Users/mkhan96/Documents/Spring2018/SEIS737/MongoDB/students18.csv --headerline

db.temp.aggregate([{$group:{_id: "$Name", Score: {$push: "$Score"}}}, {$out : "students"}])
//To push to students table with scores in an array (collection) and Name as _id
How many students are in the collection?
Script: var count = db.students.count()
Result:
>count
27

Show the name and the scores for all students who have the same score for, at least, two of their three tests.
Script:
db.students.find({ $where: "new Set(this.Score).size <= 2" });
Result:
```
{ "_id" : "Mike", "Score" : [ 86, 99, 99 ] }
{ "_id" : "Renee", "Score" : [ 86, 44, 86 ] }
{ "_id" : "Art", "Score" : [ 66, 66, 66 ] }
{ "_id" : "Sammy", "Score" : [ 77, 77, 77 ] }
{ "_id" : "Farah", "Score" : [ 86, 99, 99 ] }
{ "_id" : "Saman", "Score" : [ 76, 34, 76 ] }
{ "_id" : "Golpar", "Score" : [ 77, 87, 87 ] }
```

Show the name and the scores for all students who have the same score for all of their three tests.
Script:
> db.students.find({ $where: "new Set(this.Score).size === 1" });

Result:
```
{ "_id" : "Art", "Score" : [ 66, 66, 66 ] }
{ "_id" : "Sammy", "Score" : [ 77, 77, 77 ] }
```

Print the number of students who have the same score for ONLY two of their three tests
Script:

```
var countNamesWithTwoSameScores  = db.students.find( { $where: "new Set(this.Score).size
=== 2" } );
Result:
{ "_id" : "Mike", "Score" : [ 86, 99, 99 ] }
{ "_id" : "Renee", "Score" : [ 86, 44, 86 ] }
{ "_id" : "Farah", "Score" : [ 86, 99, 99 ] }
{ "_id" : "Saman", "Score" : [ 76, 34, 76 ] }
{ "_id" : "Golpar", "Score" : [ 77, 87, 87 ] }

> countNamesWithTwoSameScores.length()
5
```

Find the name and the scores for those student(s) who have the highest overall average

Script:

```
var name = "";
var maxAvg = 0.0;
var studentsWithHighestAverage = db.students.find().map(function(student) {
  var sum = 0.0;
  for( var i = 0; i < student.Score.length; i++){
    sum += parseFloat(student.Score[i]);

  }
  if(maxAvg < parseFloat(sum/student.Score.length)){
    maxAvg = (sum/student.Score.length).toFixed(3) ;
    name = student._id;
  }

  return name;
});
>  db.students.find({"_id" : studentsWithHighestAverage[0]}).pretty()
Result:
{ "_id" : "Mike", "Score" : [ 86, 99, 99 ] }

  The name of the student with maximum average: Mike and average: 94.667
```

6. Print the name of student and the effective grade for the student calculated as
the highest score multiplied by 60% plus
the middle score multiplied by 40%

For example if John has the scores 40, 70, and 80, his effective score for the class is
80 * .6 + 70 * .4 = 76

Script:

```
var studentsEffectiveScores = db.students.find().map(function(student) {
  var effectiveScore = 0.0;
  var name = "";
  student.Score.sort(function(a, b){return b - a});
  effectiveScore = parseFloat(student.Score[0] * 0.6 + student.Score[1] * 0.4 ).toFixed(3);

  return {
    name: student._id,
    effectiveScore: effectiveScore
  };
});
```

Result:

```
> studentsEffectiveScores
[
	{
		"name" : "Mike",
		"effectiveScore" : "99.000"
	},
	{
		"name" : "Farouq",
		"effectiveScore" : "65.800"
	},
	{
		"name" : "Ed",
		"effectiveScore" : "85.200"
	},
	{
		"name" : "Poneh",
		"effectiveScore" : "90.400"
	},
	{
		"name" : "Kurt",
		"effectiveScore" : "71.200"
	},
	{
		"name" : "Susi",
		"effectiveScore" : "71.800"
	},
	{
		"name" : "Renee",
		"effectiveScore" : "86.000"
	},
	{
		"name" : "Roger",
		"effectiveScore" : "80.600"
	},
	{
		"name" : "Tammy",
		"effectiveScore" : "88.400"
	},
	{
		"name" : "Mo",
		"effectiveScore" : "47.800"
	},
	{
		"name" : "Art",
		"effectiveScore" : "66.000"
	},
	{
		"name" : "Sammy",
		"effectiveScore" : "77.000"
	},
	{
		"name" : "Josh",
		"effectiveScore" : "83.200"
	},
	{
		"name" : "April",
		"effectiveScore" : "82.000"
	},
	{
		"name" : "Jordan",
		"effectiveScore" : "62.000"
	},
	{
		"name" : "Elaine",
		"effectiveScore" : "90.200"
	},
	{
		"name" : "Joseph",
		"effectiveScore" : "56.800"
	},
	{
		"name" : "LeeAnn",
		"effectiveScore" : "54.600"
	},
	{
		"name" : "Monir",
		"effectiveScore" : "84.200"
	},
	{
			"name" : "Joe",
			"effectiveScore" : "68.000"
		},
		{
			"name" : "Lynn",
			"effectiveScore" : "73.200"
		},
		{
			"name" : "Shah",
			"effectiveScore" : "67.600"
		},
		{
			"name" : "Sam",
			"effectiveScore" : "79.400"
		},
		{
			"name" : "Farah",
			"effectiveScore" : "99.000"
		},
		{
			"name" : "Saman",
			"effectiveScore" : "76.000"
		},
		{
			"name" : "Golpar",
			"effectiveScore" : "87.000"
		},
		{
			"name" : "Saeed",
			"effectiveScore" : "88.800"
		}
]
```

B. Repeat above six steps using Aggregation Framework.

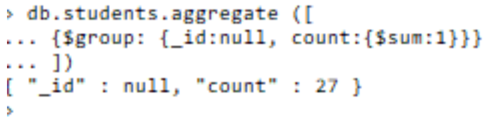
Set Up:
Importing data

```
C:\mongo-HW>mongoimport -d test  -c temp --type csv --port 27017 --file C:\mongo-HW\students18.csv --headerline
db.temp.aggregate( [{$group:{_id: "$Name", Score: {$push: "$Score"}}}, {$out : "students"}])
```

Q1:
Script:

```
db.students.aggregate ([
{$group: {_id:null, count:{$sum:1}}}
```

])
Result:



```
> db.students.aggregate ([
... {$group: {_id:null, count:{$sum:1}}}
... ])
{ "_id" : null, "count" : 27 }
>
```

Q2.
mongoimport --db students --collection temp1 --type csv --file /Users/ /Documents/Spring2018/SEIS737/MongoDB/students18.csv --headerline

Script:
```
db.temp1.aggregate({ $group: { _id: "$Name", Score: { $push: "$Score" }, scoreDuplicate: { $addToSet: "$Score" } } },{ $match: {$nor: [{ scoreDuplicate: {$size: 3}}]}   })
```
Result:
```
{ "_id" : "Mike", "Score" : [ 86, 99, 99 ], "scoreDuplicate" : [ 99, 86 ] }
{ "_id" : "Renee", "Score" : [ 86, 44, 86 ], "scoreDuplicate" : [ 44, 86 ] }
{ "_id" : "Art", "Score" : [ 66, 66, 66 ], "scoreDuplicate" : [ 66 ] }
{ "_id" : "Sammy", "Score" : [ 77, 77, 77 ], "scoreDuplicate" : [ 77 ] }
{ "_id" : "Farah", "Score" : [ 86, 99, 99 ], "scoreDuplicate" : [ 99, 86 ] }
{ "_id" : "Saman", "Score" : [ 76, 34, 76 ], "scoreDuplicate" : [ 34, 76 ] }
{ "_id" : "Golpar", "Score" : [ 87, 77, 87 ], "scoreDuplicate" : [ 77, 87 ] }
>
```

Q3.
Script:
```
> db.temp1.aggregate({ $group: { _id: "$Name", Score: { $push: "$Score" }, scoreDuplicate: { $addToSet: "$Score" } } },{ $match: { "scoreDuplicate": { $size: 1 } } })
```
Result:
```
{ "_id" : "Art", "Score" : [ 66, 66, 66 ], "scoreDuplicate" : [ 66 ] }
{ "_id" : "Sammy", "Score" : [ 77, 77, 77 ], "scoreDuplicate" : [ 77 ] }
```

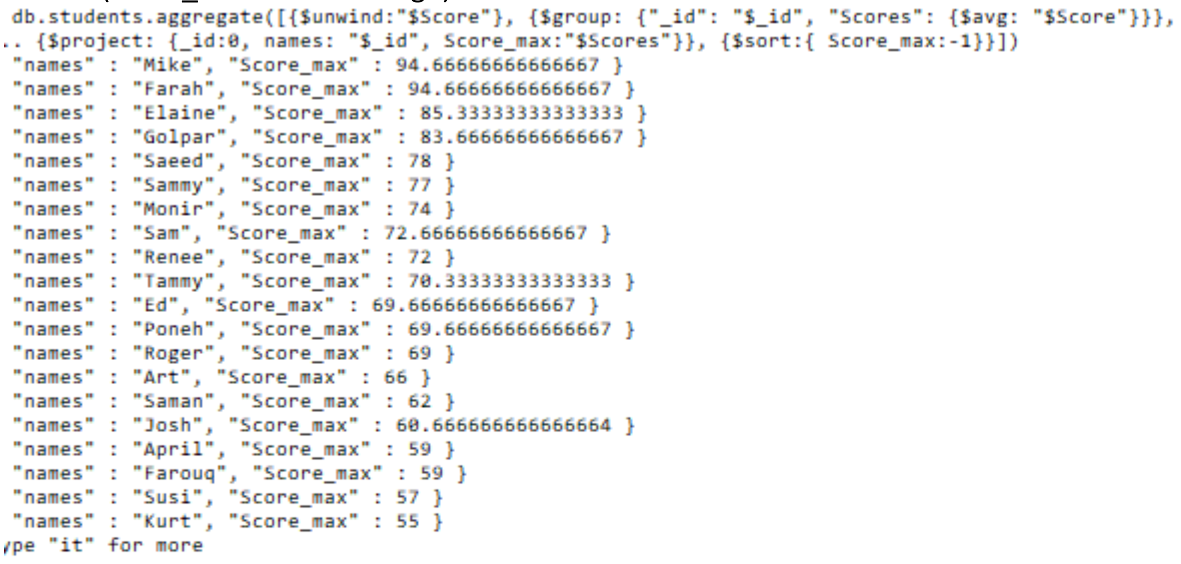
Q4:
Script:

```
> db.temp1.aggregate({ $group: { _id: "$Name", Score: { $push: "$Score" }, scoreDuplicate: { $addToSet: "$Score" } } },{ $match: { "scoreDuplicate": { $size: 2 } } })
```
Result:
```
{ "_id" : "Mike", "Score" : [ 86, 99, 99 ], "scoreDuplicate" : [ 99, 86 ] }
{ "_id" : "Renee", "Score" : [ 86, 44, 86 ], "scoreDuplicate" : [ 44, 86 ] }
{ "_id" : "Farah", "Score" : [ 86, 99, 99 ], "scoreDuplicate" : [ 99, 86 ] }
{ "_id" : "Saman", "Score" : [ 76, 34, 76 ], "scoreDuplicate" : [ 34, 76 ] }
{ "_id" : "Golpar", "Score" : [ 87, 77, 87 ], "scoreDuplicate" : [ 77, 87 ] }
```

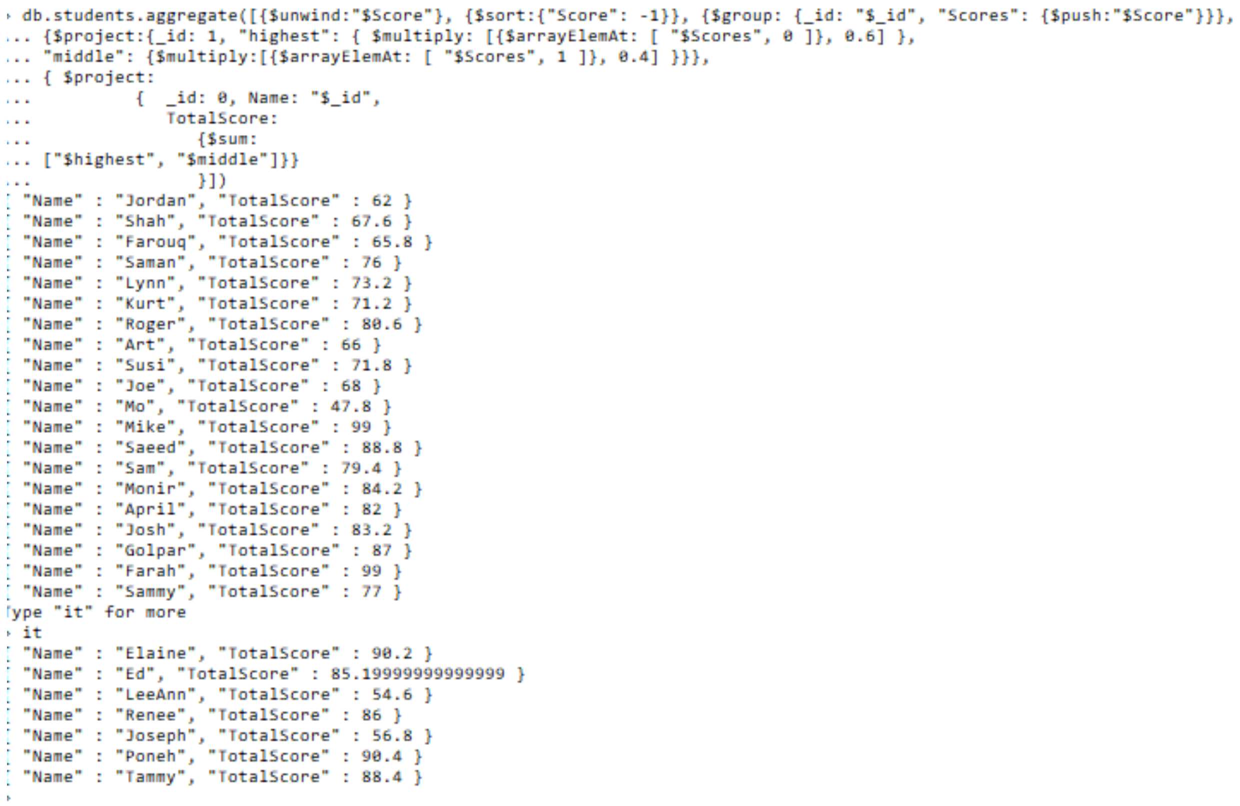
So, the count is 5.

Q.5
Script:
db.temp1.aggregate([{$group:{_id: "$Name", Score: {$push: "$Score"}, average: { $avg: "$Score" } }}, {$out : "averageScore"}])

db.averageScore.find().sort({"$Score ":-1}).limit(1)
Result:
{ "_id" : "Mike", "Score" : [86, 99, 99], "average" : 94.66666666666667 }

Another way:
Script:
db.students.aggregate([{$unwind:"$Score"}, {$group: {"_id": "$_id", "Scores": {$avg: "$Score"}}},
{$project: {_id:0, names: "$_id", Score_max:"$Scores"}}, {$sort:{ Score_max:-1}}])
Result: (Score_max is the average)

```
 db.students.aggregate([{$unwind:"$Score"}, {$group: {"_id": "$_id", "Scores": {$avg: "$Score"}}},
.. {$project: {_id:0, names: "$_id", Score_max:"$Scores"}}, {$sort:{ Score_max:-1}}])
 "names" : "Mike", "Score_max" : 94.66666666666667 }
 "names" : "Farah", "Score_max" : 94.66666666666667 }
 "names" : "Elaine", "Score_max" : 85.33333333333333 }
 "names" : "Golpar", "Score_max" : 83.66666666666667 }
 "names" : "Saeed", "Score_max" : 78 }
 "names" : "Sammy", "Score_max" : 77 }
 "names" : "Monir", "Score_max" : 74 }
 "names" : "Sam", "Score_max" : 72.66666666666667 }
 "names" : "Renee", "Score_max" : 72 }
 "names" : "Tammy", "Score_max" : 70.33333333333333 }
 "names" : "Ed", "Score_max" : 69.66666666666667 }
 "names" : "Poneh", "Score_max" : 69.66666666666667 }
 "names" : "Roger", "Score_max" : 69 }
 "names" : "Art", "Score_max" : 66 }
 "names" : "Saman", "Score_max" : 62 }
 "names" : "Josh", "Score_max" : 60.666666666666664 }
 "names" : "April", "Score_max" : 59 }
 "names" : "Farouq", "Score_max" : 59 }
 "names" : "Susi", "Score_max" : 57 }
 "names" : "Kurt", "Score_max" : 55 }
ype "it" for more
```

Q6.
Script:
```
db.students.aggregate([{$unwind:"$Score"}, {$sort:{"Score": -1}}, {$group: {_id: "$_id",
"Scores": {$push:"$Score"}}},
{$project:{_id: 1, "highest": { $multiply: [{$arrayElemAt: [ "$Scores", 0 ]}, 0.6] },
                  "middle": {$multiply:[{$arrayElemAt: [ "$Scores", 1 ]}, 0.4] }}},
{ $project:
      {  _id: 0, Name: "$_id",
        TotalScore:
          {$sum:
                  ["$highest", "$middle"]}}
          }])
```
Result:



```
> db.students.aggregate([{$unwind:"$Score"}, {$sort:{"Score": -1}}, {$group: {_id: "$_id", "Scores": {$push:"$Score"}}},
... {$project:{_id: 1, "highest": { $multiply: [{$arrayElemAt: [ "$Scores", 0 ]}, 0.6] },
... "middle": {$multiply:[{$arrayElemAt: [ "$Scores", 1 ]}, 0.4] }}},
... { $project:
...          {  _id: 0, Name: "$_id",
...             TotalScore:
...                {$sum:
... ["$highest", "$middle"]}}
...                }])
{ "Name" : "Jordan", "TotalScore" : 62 }
{ "Name" : "Shah", "TotalScore" : 67.6 }
{ "Name" : "Farouq", "TotalScore" : 65.8 }
{ "Name" : "Saman", "TotalScore" : 76 }
{ "Name" : "Lynn", "TotalScore" : 73.2 }
{ "Name" : "Kurt", "TotalScore" : 71.2 }
{ "Name" : "Roger", "TotalScore" : 80.6 }
{ "Name" : "Art", "TotalScore" : 66 }
{ "Name" : "Susi", "TotalScore" : 71.8 }
{ "Name" : "Joe", "TotalScore" : 68 }
{ "Name" : "Mo", "TotalScore" : 47.8 }
{ "Name" : "Mike", "TotalScore" : 99 }
{ "Name" : "Saeed", "TotalScore" : 88.8 }
{ "Name" : "Sam", "TotalScore" : 79.4 }
{ "Name" : "Monir", "TotalScore" : 84.2 }
{ "Name" : "April", "TotalScore" : 82 }
{ "Name" : "Josh", "TotalScore" : 83.2 }
{ "Name" : "Golpar", "TotalScore" : 87 }
{ "Name" : "Farah", "TotalScore" : 99 }
{ "Name" : "Sammy", "TotalScore" : 77 }
Type "it" for more
> it
{ "Name" : "Elaine", "TotalScore" : 90.2 }
{ "Name" : "Ed", "TotalScore" : 85.19999999999999 }
{ "Name" : "LeeAnn", "TotalScore" : 54.6 }
{ "Name" : "Renee", "TotalScore" : 86 }
{ "Name" : "Joseph", "TotalScore" : 56.8 }
{ "Name" : "Poneh", "TotalScore" : 90.4 }
{ "Name" : "Tammy", "TotalScore" : 88.4 }
>
```