Exercise 1

Data selection:

```
db.student_performance.findOne()
```

Description:
Here we are selecting only the first data.

Data projection:

```
db.student_performance.find({}, { "gender": 1, "math score": 1, "reading score": 1, "_id": 0 })
```

Description:
In data, we are supposed to only project relevant fields for all documents. As a result we have passed the first argument as {} which means no filter applied. Whichever field = "1" is supposed to be displayed. Whichever field = "0" should not be displayed.

Data filtering:

```
db.student_performance.find({ "math score": { $gt: 80 }, "reading score":  {$gt: 80}, "writing score": {$gt: 80}});
```

Description:
Here we are filtering documents who has reading score > 80, math score > 80 and writing score > 80

Data Transformation:

```
db.student_performance.aggregate([
 {
 $project: {
_id: 1,
 averageScore: {
 $avg: [
"$math score",
"$reading score",
"$writing score"
]}}}])
```

Description:
The following query in MongoDB calculates each student's average score in the ‘student_performance’ collection. It contains the ‘$project’ stage which reshapes the documents by including each student’s ‘_id’ as well as a ’averageScore’ field. The average is generated by the use of the ‘$avg’ operator on the math, reading and writing scores. What results is a student database, bearing a unique ‘_id’ and an average score for each of them.

To check the importance of test preparation key field:

```
db.student_performance.aggregate([
 {
  $match: {
   'test preparation course': 'completed', // Filtering for students who completed the
course
  }
 },
 {
  $count: "completed" // Count the number of matching documents
 }
])
```

```
db.student_performance.aggregate([
 {
  $match: {
    'test preparation course': 'completed', // Filter for students who completed the
course
  }
 },
 {
  $project: {
   _id: 1,
   averageScore: {
    $avg: [
     "$math score",
     "$reading score",
     "$writing score"
    ]
   }
  }
 },
 {
  $match: {
   averageScore: { $gte: 80 } // Filter for students with average score 80 and above
  }
 },
 {
  $count: "studentsWithHighScores" // Count the number of matching documents
 }
])
```

```
db.student_performance.aggregate([
 {
  $match: {
   'test preparation course': 'completed', // Filter for students who completed the
course
  }
 },
 {
  $facet: {
   totalCount: [
    { $count: "count" } // Count total students who completed the course
   ],
   highScoreCount: [
    {
     $project: {
      averageScore: {
       $avg: [
        "$math score",
        "$reading score",
        "$writing score"
       ]
      }
     }
    },
    {
     $match: {
      averageScore: { $gte: 80 } // Match students with average score >= 80
     }
    },
    { $count: "count" } // Count those students
   ]
  }
 },
 {
  $project: {
   percentage: {
    $multiply: [
     { $divide: [
       { $arrayElemAt: ["$highScoreCount.count", 0] }, // Count of high scorers
       { $arrayElemAt: ["$totalCount.count", 0] } // Total count of students
      ] },
     100 // Convert to percentage
    ]
   }
  }
 }
```

])

(We have similarly executed the queries for "none" value.)

Description:
The assertion implies that all the students who underwent the test preparation course performed better than all those who did not, which indicates that the course was helpful. Apart from that, by composing MongoDB queries to identify the students who completed the course and find their mean scores, it is possible to evaluate also these amounts. The comparison of these scores shows the positive effect of the course. Further reinforcing the notion that the course led to better performance is the calculation of the proportion of students who scored above 80 who attended the course. It can, therefore, be concluded from the results obtained that the course in question had an impact on students' performance.

Data Combination/Grouping:

```
db.student_performance.aggregate([
 {
  $group: {
   _id: "$math score", // Group by math score
   averageReadingScore: { $avg: "$reading score" },
   averageWritingScore: { $avg: "$writing score" }
  }
 }
])
```

Description:
The MongoDB aggregation query in question collects the students data based on the math scores, and performs an average of the reading and writing scores for each of the math score groups. The query makes use of the group stage in which the id is defined as the available unique mathematical scores. It specifies the average reading score and the average writing score for the students in that group using the average operator. The outcome is a set of documents where each document displays a particular math score with the average reading and writing scores of the students who obtained that math score. This makes it possible to study the relationship between math performance and reading and writing scores.

Grouping (student score average):

```
db.student_performance.aggregate([
 {
  $group: {
   _id: "$gender", // Group by gender
   averageMathScore: { $avg: "$math score" },
   averageReadingScore: { $avg: "$reading score" },
   averageWritingScore: { $avg: "$writing score" }
  }}])
```

Description:
The given MongoDB aggregation query clusters the student information by gender and determines the mean scores of all the subjects. At the `$group` stage a gender is defined as the `_id` and hence a breakdown of results into male students and female students is performed. For every gender group, the average of math, reading and writing scores is calculated using the `$avg` operator. The output is a collection of documents each presenting one gender and the average scores in math, reading and writing for that sex which helps conduct a gender based performance evaluation.

Grouping by Gender Based Performance:

```
db.student_performance.aggregate([
 {
  $project: {
   _id: 1,
   totalScore: {
    $add: [
     "$math score",
     "$reading score",
     "$writing score"
    ]}}}])
```

Description:
The MongoDB aggregation command given herein keeps a record of the total score for each of the students by adding the scores which they managed to obtain in the different sections: math, reading, and writing. It uses the `$project` stage to include that student's `_id` as well as create a new field called `totalScore` whose value is computed using the `$add` operator. Thus, the resulting set of documents will contain the student's `_id ` and the respective total score of the student making it convenient to evaluate the complete academic performance of each student.