2. Projection Queries

a. Select Specific Fields

Project only the `name` and `gpa` fields for all students:

```
db.students.find({}, { name: 1, gpa: 1, _id: 0
});
```

b. Exclude a Field

Exclude the `age` field from the results:

```
db.students.find({}, { age: 0 });
```

c. Combine Filters and Projection

Get only students with a `gpa` greater than 3.5 and project their `name` and `gpa`:

```
db.students.find({ gpa: { $gt: 3.5 } }, { name:
1, gpa: 1, _id: 0 });
```

3. Aggregation Queries

a. Group by Department and Calculate Average GPA

Find the average `gpa` for each department:

```
db.students.aggregate([
  { $group: { _id: "$department", averageGPA: {
$avg: "$gpa" } } }
]);
```

b. Count Students by Department

Count the number of students in each department:

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

c. Filter and Then Group

Find the average `gpa` of students in the "Computer Science" department:

```
db.students.aggregate([
  { $match: { department: "Computer Science" }
},
  { $group: { _id: "$department", averageGPA: {
$avg: "$gpa" } } }
]);
```

d. Sort by GPA in Descending Order

Sort the students by their `gpa` in descending order:

```
db.students.aggregate([
  { $sort: { gpa: -1 } }
]);
```

e. Add a Custom Field

Add a new field `status` to indicate if a student is "Pass" or "Fail" based on `gpa`:

```
db.students.aggregate([
  {
    $addFields: {
      status: {
        $cond: { if: { $gte: ["$gpa", 3.5] },
then: "Pass", else: "Fail" }
      }
    }
  }
]);
```

f. Count Students Who Passed

Count the number of students with a `gpa` of 3.5 or higher:

```
db.students.aggregate([
  { $match: { gpa: { $gte: 3.5 } } },
  { $count: "passedStudents" }
]);
```

4. Combined Projection and Aggregation

a. Project and Rename Fields

Show `name` as `studentName` and `gpa` as `studentGPA`:

```
db.students.aggregate([
  {
    $project: {
      studentName: "$name",
      studentGPA: "$gpa",
      _id: 0
    }
  }
]);
```

b. Filter, Group, and Project

Find students older than 20, group by `department`, and show the average GPA with renamed fields:

```
db.students.aggregate([
  { $match: { age: { $gt: 20 } } },
  { $group: { _id: "$department", avgGPA: {
$avg: "$gpa" } } },
  {
    $project: {
      department: "$_id",
      averageGPA: "$avgGPA",
      _id: 0
```

```
        }
    }
]);
```