1) Using MongoDB

i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)).

```
use student2;

db.createCollection("Student");
```

ii) Insert required documents to the collection.

```
> db.Student.insert({_id:1,Name: "Arun", sem:"V",dept: "CSE",CGPA: 8.2,hobbies: ['cycling','swimming']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name: "Ananya", sem:"VII",dept: "ECE",CGPA: 6.8,hobbies: ['knitting','reading novels']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,Name: "Bhuvan", sem:"III",dept: "ME",CGPA: 8.8,hobbies: ['chess','collecting coins']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,Name: "Ajay", sem:"VII",dept: "CSE",CGPA: 9.1,hobbies: ['playing','reading novels']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,Name: "Colin", sem:"V",dept: "CSE",CGPA: 7.1,hobbies: ['playing','watching TV']});
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.find();
{ "_id" : 1, "Name" : "Arun", "sem" : "V", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling", "swimming" ] }
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
{ "_id" : 5, "Name" : "Colin", "sem" : "V", "dept" : "CSE", "CGPA" : 7.1, "hobbies" : [ "playing", "watching TV" ] }
>
```

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.

```
>
db.Student.aggregate({$match:{dept:"CSE"}},{$group:{_id:"$sem",AverageCGPA:{$avg:"$CGPA"}}},{$match:{AverageCGPA:{$gt:7.5}}});
{ "_id" : "VII", "AverageCGPA" : 9.1 }
{ "_id" : "V", "AverageCGPA" : 7.6499999999999995 }
```

```
> db.Student.aggregate({$match:{dept:"CSE"}},{$group:{_id:"$sem",AverageCGPA:{$avg:"$CGPA"}}},{$match:{AverageCGPA:{$gt:7.5}}});
{ "_id" : "V", "AverageCGPA" : 7.6499999999999995 }
{ "_id" : "VII", "AverageCGPA" : 9.1 }
```

iv) Insert the document for “Bhuvan” in to the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his Hobbies to “Skating”) Use “Update else insert” (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

```
> db.Student.update({_id: 3,Name:"Bhuvan"},{$set:{ Hobbies:"Skating"}},{upsert:true}
);
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

v)To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

```
> db.Student.find({},{name:1,sem: 1, _id:0});
{ "sem" : "V" }
{ "sem" : "VII" }
{ "sem" : "III" }
{ "sem" : "VII" }
{ "sem" : "V" }
```

vi) To find those documents where the Grade is set to 'VII'

```
> db.Student.find({sem:{$eq:"VII"}});
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
```

vii)To find those documents from the Students collection where the Hobbies is set to either 'Chess' or is set to 'Skating'.

```
> db.Student.find({Hobbies:{$in:['Chess','Skating']}});
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
```

viii)To find documents from the Students collection where the StudName begins with "B"

```
> db.Student.find({Name: /^B/});
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
```

ix) To find the number of documents in the Students collection.

```
> db.Student.count();
5
```

x) To sort the documents from the Students collection in the descending order of StudName.

```
> db.Student.find().sort({Name: -1});
{ "_id" : 5, "Name" : "Colin", "sem" : "V", "dept" : "CSE", "CGPA" : 7.1, "hobbies" : [ "playing", "watching TV" ] }
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
{ "_id" : 1, "Name" : "Arun", "sem" : "V", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling", "swimming" ] }
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
```

xi) Command used to export MongoDB JSON documents from "Student" Collection into the "Students" database into a CSV file "Output.txt".

```
> mongoexport --host localhost --db studentDB --collection Student --csv --out /Downloads/student.txt -fields "Name","sem"
uncaught exception: SyntaxError: unexpected token: identifier :
@(shell):1:14
```