SWEN 432 assignment 4
SONGBO WU

Question 1.

```
> db.reserves.aggregate ([ {$match: {'reserves.sailor.name' :{$exists: true} }}, {$group:
{_id:'$reserves.sailor.sailorId',name :{$first:'$reserves.sailor.name'} , skills: {$first:'$reserves.sailor.skills'} ,
address : {$first:'$reserves.sailor.address'}}}]);
{ "_id" : 110, "name" : "Paul", "skills" : [ "row", "swim" ], "address" : "Upper Hutt" }
{ "_id" : 777, "name" : "Alva", "skills" : [ "row", "sail", "motor", "dance" ], "address" : "Masterton" }
{ "_id" : 919, "name" : "Eileen", "skills" : [ "sail", "motor", "swim" ], "address" : "Lower Hutt" }
{ "_id" : 111, "name" : "Peter", "skills" : [ "row", "sail", "motor" ], "address" : "Upper Hutt" }
{ "_id" : 999, "name" : "Charmain", "skills" : [ "row" ], "address" : "Upper Hutt" }
{ "_id" : 818, "name" : "Milan", "skills" : [ "row", "sail", "motor", "first aid" ], "address" : "Wellington" }
{ "_id" : 707, "name" : "James", "skills" : [ "row", "sail", "motor", "fish" ], "address" : "Wellington" }
>
```

Question 2.

```
> db.reserves.aggregate ([  {$match: {'reserves.sailor.sailorId':  {$exists:true}}} ,
 {$group : {_id : '$reserves.sailor.sailorId',  "name": {"$first":'$reserves.sailor.name'} ,   "address":
{"$first":'$reserves.sailor.address'} ,  "no_of_reserves" : {$sum : 1} }} ,
{$sort : {"no_of_reserves" : -1}} ,
{$limit : 1}]);

{ "_id" : 818, "name" : "Milan", "address" : "Wellington", "no_of_reserves" : 6 }
```

Question 3.

```
> db.reserves.aggregate ([  {$match: {'reserves.date':  {$exists:true}}} ,
{$group : {_id : null,    "total_reserves" : {$sum : 1} }} ,
{$sort : {"total_reserves" : -1}},
{$project:{_id:0 , "total_reserves" :1} }

]);
{ "total_reserves" : 18 }
```

Question 4.

```
> db.reserves.aggregate ([  {$match: {'reserves.date':  {$exists:true}}} ,
... ...  {$group : {_id : '$reserves.sailor.sailorId',   "no_of_reserves" : {$sum : 1} }} ,
... ...   {$group : {_id : null  ,    avg_reserves: {$avg:"$no_of_reserves"} }} ,
... ... {$sort : {"no_of_reserves" : -1}} ,
... ... {$project:{_id:0 , avg_reserves :1}}
... ... ]);
{ "avg_reserves" : 3 }
```

Question 5.

```
> var skills = db.reserves.distinct ('reserves.sailor.skills', {'reserves.sailor.name': "Paul"}) ;
>
> var myArray = [];
> db.reserves.aggregate ([  {$match: { 'reserves.boat.driven_by' :  {$exists:true}  }} ,
...  {$match: { 'reserves.boat.driven_by':  {"$not": {"$elemMatch": {"$nin" : skills }}}   }} ,
...  {$project: { boat:  '$reserves.boat.name' , _id:0 }}]).forEach(function(row){
...    myArray.push(row.boat)
...  });
>
> myArray.filter ( function (value, index, self) {  return self.indexOf(value) === index;}) ;
[ "Killer Whale", "Night Breeze", "Penguin", "Sea Gull" ]
```

```
>  //Bonus solution version 1 :
>  var av  =  db.reserves.aggregate ([  {$match: {'reserves.sailor.sailorId':  {$exists:true}}} ,
... {$group : {_id : '$reserves.sailor.sailorId',   "no_of_reserves" : {$sum : 1} }} ,
...   {$group : {_id : null  ,    avg_reserves: {$avg:"$no_of_reserves"} }} ,
...  {$sort : {"no_of_reserves" : -1}} ,
... {$project:{_id:0 , avg_reserves :1}}
...   ]).map( function(u) { return u.avg_reserves } );
>
>
>   db.reserves.aggregate ([  {$match: {'reserves.sailor.sailorId':  {$exists:true}}} ,
...  {$group : {_id : '$reserves.sailor.sailorId', name:{$first:'$reserves.sailor.name'} , "no_of_reserves" : {$sum :
1}  }} ,
...  {$match : { no_of_reserves: {$gt: parseFloat(av)} }} ,
...  {$sort : {"no_of_reserves" : -1}}
...  ]);
{ "_id" : 818, "name" : "Milan", "no_of_reserves" : 6 }
{ "_id" : 111, "name" : "Peter", "no_of_reserves" : 3 }
{ "_id" : 707, "name" : "James", "no_of_reserves" : 3 }
>
>
>
>
>
>  //Bonus solution version 2 :
>   var av  =  db.reserves.aggregate ([  {$match: {'reserves.sailor.sailorId':  {$exists:true}}} ,
...   {$group : {_id : '$reserves.sailor.sailorId',   "no_of_reserves" : {$sum : 1} }} ,
...   {$group : {_id : null  ,    avg_reserves: {$avg:"$no_of_reserves"} }} ,
...   {$sort : {"no_of_reserves" : -1}} ,
...  {$project:{_id:0 , avg_reserves :1}}
...  ]).next();
>
>
>   db.reserves.aggregate ([  {$match: {'reserves.sailor.sailorId':  {$exists:true}}} ,
...  {$group : {_id : '$reserves.sailor.sailorId', name:{$first:'$reserves.sailor.name'} , "no_of_reserves" : {$sum :
1}  }} ,
...  {$match : { no_of_reserves: {$gt: av.avg_reserves} }} ,
...  {$sort : {"no_of_reserves" : -1}}
...  ]);
{ "_id" : 818, "name" : "Milan", "no_of_reserves" : 6 }
{ "_id" : 111, "name" : "Peter", "no_of_reserves" : 3 }
{ "_id" : 707, "name" : "James", "no_of_reserves" : 3 }
```

Question 6.

a)

based on ranges partitioning

b)
i.

For 2 shards ,   6
For 5 shards , 2
For 10 shards , 1.2

ii.

```
sha-mongo stop ;sha-mongo cleanall ;sha-mongo init 2 ; sha-mongo test ;    sha-mongo status ;
{ "user_id" : 48999 } -->> { "user_id" : 59999 } on : shard0001 Timestamp(6, 2)
chunks:
                    shard0000      6
                    shard0001      5
```

it belongs to shard0001.

```
sha-mongo stop ;sha-mongo cleanall ;sha-mongo init 5 ; sha-mongo test ;    sha-mongo status ;
{ "user_id" : 48999 } -->> { "user_id" : 58999 } on : shard0004 Timestamp(6, 1)
chunks:
                      shard0002      2
                      shard0000      2
                      shard0001      2
                      shard0003      2
                      shard0004      2
```

it belongs to shard0004.

```
sha-mongo stop ;sha-mongo cleanall ;sha-mongo init 10 ; sha-mongo test ;    sha-mongo status ;
{ "user_id" : 45999 } -->> { "user_id" : 55999 } on : shard0005 Timestamp(7, 1)

chunks:
                        shard0000      2
                        shard0001      2
                        shard0002      1
                        shard0003      1
                        shard0004      1
                        shard0005      1
                        shard0006      1
                        shard0007      1
                        shard0008      1
                        shard0009      1
```

it belongs to shard0005.

c)

```
arthur: [A4] % sha-mongo connect 5 ;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27025/test
> use mydb ;
switched to db mydb
> db.user.find ({user_id:55555});
{ "_id" : ObjectId("5920102bdec48588e3d182ac"), "user_id" : 55555, "name" : "Bill", "number" : 9349 }
> db.user.find ({user_id:1});
```

d)

```
arthur: [A4] % sha-mongo connect;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27017/test
mongos> use mydb ;
switched to db mydb
mongos>  db.user.find ({user_id:55555});
{ "_id" : ObjectId("5920102bdec48588e3d182ac"), "user_id" : 55555, "name" : "Bill", "number" : 9349 }
mongos> db.user.find ({user_id:1});
{ "_id" : ObjectId("59201029dec48588e3d0a9aa"), "user_id" : 1, "name" : "George", "number" : 894 }
```

e)

Queries runs on mongos will search records accross all the replication sets/shards.
f)

```
arthur: [A4] % sha-mongo stop 5 ;
Stopping mongod --port 27025 shadb5
```

```
arthur: [A4] % sha-mongo connect ;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27017/test
mongos> db.user.find ({user_id:55555});
mongos> ^C
bye
arthur: [A4] % sha-mongo connect ;
```

```
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27017/test
mongos> use mydb;
switched to db mydb
mongos> db.user.find ({user_id:55555});
error: {
        "$err" : "socket exception [CONNECT_ERROR] for 127.0.0.1:27025",
        "code" : 11002,
        "shard" : "shard0005"
}
mongos> db.user.find ({user_id:1});
{ "_id" : ObjectId("59201029dec48588e3d0a9aa"), "user_id" : 1, "name" : "George", "number" : 894 }
```

10% became unavailable.

```
arthur: [A4] % sha-mongo start  ;
mongos> db.user.find ({user_id:55555});
{ "_id" : ObjectId("5920102bdec48588e3d182ac"), "user_id" : 55555, "name" : "Bill", "number" : 9349 }
```

It becomes available again when start the sha-mongo .

Question 7

a)

```
arthur: [A4] % sharep-mongo init ;sharep-mongo test ;
```

```
arthur: [A4] % sharep-mongo connect 0 0
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27020/test
rs0:PRIMARY> rs.status()
{
        "set" : "rs0",
        "date" : ISODate("2017-05-20T10:26:43Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "127.0.0.1:27020",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 935,
                        "optime" : Timestamp(1495275069, 1),
                        "optimeDate" : ISODate("2017-05-20T10:11:09Z"),
                        "electionTime" : Timestamp(1495275078, 1),
                        "electionDate" : ISODate("2017-05-20T10:11:18Z"),
                        "self" : true
                },
                {
                        "_id" : 1,
                        "name" : "127.0.0.1:27021",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 933,
                        "optime" : Timestamp(1495275069, 1),
                        "optimeDate" : ISODate("2017-05-20T10:11:09Z"),
                        "lastHeartbeat" : ISODate("2017-05-20T10:26:42Z"),
                        "lastHeartbeatRecv" : ISODate("2017-05-20T10:26:42Z"),
                        "pingMs" : 0,
                        "syncingTo" : "127.0.0.1:27020"
                },
                {
                        "_id" : 2,
                        "name" : "127.0.0.1:27022",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 933,
                        "optime" : Timestamp(1495275069, 1),
                        "optimeDate" : ISODate("2017-05-20T10:11:09Z"),
                        "lastHeartbeat" : ISODate("2017-05-20T10:26:42Z"),
                        "lastHeartbeatRecv" : ISODate("2017-05-20T10:26:42Z"),
                        "pingMs" : 0,
                        "syncingTo" : "127.0.0.1:27020"
                }
        ],
        "ok" : 1
}
rs0:PRIMARY>
```

So the port number of the master server of the replica set rs0 is 27020

b)

i.

```
arthur: [A4] % sharep-mongo connect 0 0 ;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27020/test
rs0:PRIMARY> show dbs ;
admin  (empty)
local  0.281GB
mydb   0.031GB
rs0:PRIMARY> use mydb;
switched to db mydb
rs0:PRIMARY> db.user.find({user_id:1})
{ "_id" : ObjectId("59201a50ff759c038bf6c11f"), "user_id" : 1, "name" : "Eliot", "number" : 1723 }
```

ii.

```
rs0:PRIMARY> db.user.insert({"user_id": 100000, "name": "Steve", "number": 0});
WriteResult({ "nInserted" : 1 })
```

c)

```
arthur: [A4] % sharep-mongo connect 0 1 ;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27021/test
rs0:SECONDARY> use mydb;
switched to db mydb
rs0:SECONDARY> db.user.find({user_id:1});
error: { "$err" : "not master and slaveOk=false", "code" : 13435 }
rs0:SECONDARY> db.user.insert({"user_id": 100000, "name": "Steve", "number": 0});
WriteResult({ "writeError" : { "code" : undefined, "errmsg" : "not master" } })
rs0:SECONDARY>
```

d)

i.

```
arthur: [A4] % sharep-mongo stop 0 0 ;
Stopping mongod --port 27020 sharep-rs0-0
  status is   "stateStr" : "(not reachable/healthy)",
```

ii.

```
arthur: [A4] % sharep-mongo connect ;
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27017/test
mongos> use mydb ;
```

```
switched to db mydb
mongos> db.user.find({user_id:1});
{ "_id" : ObjectId("59201a50ff759c038bf6c11f"), "user_id" : 1, "name" : "Eliot", "number" : 1723 }
mongos> db.user.insert({"user_id": 100000, "name": "Steve", "number": 0});
WriteResult({ "nInserted" : 1 })
```

e)

i.

```
arthur: [A4] % sharep-mongo connect 0 2
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27022/test
rs0:SECONDARY> rs.status()
{
	"set" : "rs0",
	"date" : ISODate("2017-05-20T10:44:09Z"),
	"myState" : 2,
	"members" : [
		{
			"_id" : 0,
			"name" : "127.0.0.1:27020",
			"health" : 0,
			"state" : 8,
			"stateStr" : "(not reachable/healthy)",
			"uptime" : 0,
			"optime" : Timestamp(1495276311, 1),
			"optimeDate" : ISODate("2017-05-20T10:31:51Z"),
			"lastHeartbeat" : ISODate("2017-05-20T10:44:04Z"),
			"lastHeartbeatRecv" : ISODate("2017-05-20T10:39:20Z"),
			"pingMs" : 0
		},
		{
			"_id" : 1,
			"name" : "127.0.0.1:27021",
			"health" : 0,
			"state" : 8,
			"stateStr" : "(not reachable/healthy)",
			"uptime" : 0,
			"optime" : Timestamp(1495276842, 1),
			"optimeDate" : ISODate("2017-05-20T10:40:42Z"),
			"lastHeartbeat" : ISODate("2017-05-20T10:44:07Z"),
			"lastHeartbeatRecv" : ISODate("2017-05-20T10:43:16Z"),
			"pingMs" : 0,
                        "syncingTo" : "127.0.0.1:27022"
                },
                {
                        "_id" : 2,
                        "name" : "127.0.0.1:27022",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 1980,
                        "optime" : Timestamp(1495276842, 1),
                        "optimeDate" : ISODate("2017-05-20T10:40:42Z"),
                        "self" : true
                }
        ],
        "ok" : 1
}
```

There is only one SECONDARY available in rs0

ii

```
arthur: [A4] % sharep-mongo connect
MongoDB shell version: 2.6.7
connecting to: 127.0.0.1:27017/test
mongos> use mydb;
switched to db mydb
mongos> db.user.find({user_id:1});
error: {
        "$err" : "ReplicaSetMonitor no master found for set: rs0",
        "code" : 10009,
        "shard" : "rs0"
}
mongos>
```

f)
The mongos redirects all the W/R queries to master node according to shardkey. Queries runs on mongos will search records accross all the replication sets/shards.
But queries run in the master node will only search the records in the current shard.
if there is no master in the Replica Set. The eventually consistency can not be meet thus the data in that set is not available for reading.