**mongoDB - Assignment 1**

Microsoft Windows [Version 10.0.19041.1110]

(c) Microsoft Corporation. All rights reserved.

C:\Users\Shubhangi>mongod

{"t":{"$date":"2021-08-01T20:33:48.317+05:30"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}

{"t":{"$date":"2021-08-01T20:33:48.320+05:30"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}

{"t":{"$date":"2021-08-01T20:33:48.321+05:30"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in use."}

{"t":{"$date":"2021-08-01T20:33:48.323+05:30"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":10928,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bit","host":"DESKTOP-CRLD5UE"}}

{"t":{"$date":"2021-08-01T20:33:48.323+05:30"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Target operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}

{"t":{"$date":"2021-08-01T20:33:48.323+05:30"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"4.4.7","gitVersion":"abb6b9c2bf675e9e2aeaecba05f0f8359d99e203","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86\_64","target\_arch":"x86\_64"}}}}

{"t":{"$date":"2021-08-01T20:33:48.323+05:30"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Microsoft Windows 10","version":"10.0 (build 19041)"}}}

{"t":{"$date":"2021-08-01T20:33:48.323+05:30"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{}}}

{"t":{"$date":"2021-08-01T20:33:48.324+05:30"},"s":"E", "c":"STORAGE", "id":20557, "ctx":"initandlisten","msg":"DBException in initAndListen, terminating","attr":{"error":"NonExistentPath: Data directory C:\\data\\db\\ not found. Create the missing directory or specify another path using (1) the --dbpath command line option, or (2) by adding the 'storage.dbPath' option in the configuration file."}}

{"t":{"$date":"2021-08-01T20:33:48.325+05:30"},"s":"I", "c":"REPL", "id":4784900, "ctx":"initandlisten","msg":"Stepping down the ReplicationCoordinator for shutdown","attr":{"waitTimeMillis":10000}}

{"t":{"$date":"2021-08-01T20:33:48.325+05:30"},"s":"I", "c":"COMMAND", "id":4784901, "ctx":"initandlisten","msg":"Shutting down the MirrorMaestro"}

{"t":{"$date":"2021-08-01T20:33:48.326+05:30"},"s":"I", "c":"SHARDING", "id":4784902, "ctx":"initandlisten","msg":"Shutting down the WaitForMajorityService"}

{"t":{"$date":"2021-08-01T20:33:48.327+05:30"},"s":"I", "c":"NETWORK", "id":20562, "ctx":"initandlisten","msg":"Shutdown: going to close listening sockets"}

{"t":{"$date":"2021-08-01T20:33:48.330+05:30"},"s":"I", "c":"NETWORK", "id":4784905, "ctx":"initandlisten","msg":"Shutting down the global connection pool"}

{"t":{"$date":"2021-08-01T20:33:48.331+05:30"},"s":"I", "c":"STORAGE", "id":4784906, "ctx":"initandlisten","msg":"Shutting down the FlowControlTicketholder"}

{"t":{"$date":"2021-08-01T20:33:48.332+05:30"},"s":"I", "c":"-", "id":20520, "ctx":"initandlisten","msg":"Stopping further Flow Control ticket acquisitions."}

{"t":{"$date":"2021-08-01T20:33:48.332+05:30"},"s":"I", "c":"NETWORK", "id":4784918, "ctx":"initandlisten","msg":"Shutting down the ReplicaSetMonitor"}

{"t":{"$date":"2021-08-01T20:33:48.333+05:30"},"s":"I", "c":"SHARDING", "id":4784921, "ctx":"initandlisten","msg":"Shutting down the MigrationUtilExecutor"}

{"t":{"$date":"2021-08-01T20:33:48.334+05:30"},"s":"I", "c":"CONTROL", "id":4784925, "ctx":"initandlisten","msg":"Shutting down free monitoring"}

{"t":{"$date":"2021-08-01T20:33:48.334+05:30"},"s":"I", "c":"STORAGE", "id":4784927, "ctx":"initandlisten","msg":"Shutting down the HealthLog"}

{"t":{"$date":"2021-08-01T20:33:48.335+05:30"},"s":"I", "c":"STORAGE", "id":4784929, "ctx":"initandlisten","msg":"Acquiring the global lock for shutdown"}

{"t":{"$date":"2021-08-01T20:33:48.335+05:30"},"s":"I", "c":"-", "id":4784931, "ctx":"initandlisten","msg":"Dropping the scope cache for shutdown"}

{"t":{"$date":"2021-08-01T20:33:48.336+05:30"},"s":"I", "c":"FTDC", "id":4784926, "ctx":"initandlisten","msg":"Shutting down full-time data capture"}

{"t":{"$date":"2021-08-01T20:33:48.337+05:30"},"s":"I", "c":"CONTROL", "id":20565, "ctx":"initandlisten","msg":"Now exiting"}

{"t":{"$date":"2021-08-01T20:33:48.337+05:30"},"s":"I", "c":"CONTROL", "id":23138, "ctx":"initandlisten","msg":"Shutting down","attr":{"exitCode":100}}

C:\Users\Shubhangi>mongo

MongoDB shell version v4.4.7

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("e9fa7174-15d5-4f5b-9daa-3f1e35d28282") }

MongoDB server version: 4.4.7

---

The server generated these startup warnings when booting:

2021-07-31T22:04:39.916+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

**Insert Documents**

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

> use mongo\_practice

switched to db mongo\_practice

> db.movies.insert({title:"Fight Club",writer:"Chuck Palahniuko",year:1999,actors:["Brad Pitt","Edward Norton"]})

WriteResult({ "nInserted" : 1 })

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

mongo\_practice 0.000GB

> db.movies.insert({title:"Pulp Fiction",writer:"Quentin Tarantino",year:1994,actors:["John Travolta","Uma Thurman"]})

WriteResult({ "nInserted" : 1 })

> db.movies.find().pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

> db.movies.insert({title:"Inglorious Basterds",writer:"Quentin Tarantino",year:2009,actors:["Brad Pitt","Diane Kruger","Eli Roth"]})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit:An Unexpected Journey",writer:"J.R.R. Tolkein",year:2012,franchise:"The Hobbit"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit:The Desolation of Smaug",writer:"J.R.R. Tolkein",year:2013,franchise:"The Hobbit"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit:The Battle of the Five Armies",writer:"J.R.R. Tolkein",year:2012,franchise:"The Hobbit",synopsis:"Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"Pee Wee Herman's Big Adventure"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"Avatar"})

WriteResult({ "nInserted" : 1 })

**Query/find documents**

**1. Get all documents**

> db.movies.find().pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6106ce621fb2405a3e0ef844"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"

}

{

"\_id" : ObjectId("6106d8d21fb2405a3e0ef84f"),

"title" : "Pee Wee Herman's Big Adventure"

}

{ "\_id" : ObjectId("6106d8e81fb2405a3e0ef850"), "title" : "Avatar" }

>

**2. get all documents with writer set to Quentin Tarantino**

> db.movies.find({writer:"Quentin Tarantino"}).pretty()

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6106ce621fb2405a3e0ef844"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

**3. get all documents where actors include Brad Pitt**

> db.movies.find({actors:"Brad Pitt"}).pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106ce621fb2405a3e0ef844"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

**4. get all documents with franchise set to The Hobbit**

> db.movies.find({franchise:"The Hobbit"}).pretty()

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"

}

**5. get all movies released in the 90s**

> db.movies.find({year:{$gte:1990,$lte:1999}}).pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

**6. get all movies released before the year 2000 or after 2010**

> db.movies.find({$or:[{year:{$gt:2010}},{year:{$lt:2000}}]}).pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"

}

**Update Documents**

**1,2,3.-->**

> db.movies.update({title:"The Hobbit:An Unexpected Journey"},{$set:{synopsis:"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

> db.movies.update({title:"The Hobbit:The Desolation of Smaug"},{$set:{synopsis:"The dwarves, along with the Bilbo Baggins and Gandalg Grey, continue theor quest to reclaim Erebor, their homeland, from Smaug. Bilbo baggins is in possession of a mysterious and magical ring."}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

> db.movies.update({title:"Pulp Fiction"},{$set:{actors:["Samuel L. Jackson","John Travolta","Uma Thurman"]}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.movies.find().pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"Samuel L. Jackson",

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6106ce621fb2405a3e0ef844"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with the Bilbo Baggins and Gandalf Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of the rising darkness"

}

{

"\_id" : ObjectId("6106d8d21fb2405a3e0ef84f"),

"title" : "Pee Wee Herman's Big Adventure"

}

{ "\_id" : ObjectId("6106d8e81fb2405a3e0ef850"), "title" : "Avatar" }

>

**Text Search**

**1.**

> db.movies.createIndex({synopsis:"text"})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

> db.movies.find({$text:{$search:"Bilbo"}}).pretty()

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with the Bilbo Baggins and Gandalf Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of the rising darkness"

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

**2.**

> db.movies.find({$text:{$search:"Gandalf"}}).pretty()

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with the Bilbo Baggins and Gandalf Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo baggins is in possession of a mysterious and magical ring."

}

**3.**

> db.movies.find({$text:{$search:"Bilbo -Gandalf"}}).pretty()

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

**4.**

> db.movies.find({$text:{$search:"dwarves hobbit"}}).pretty()

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with the Bilbo Baggins and Gandalf Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo baggins is in possession of a mysterious and magical ring."

}

**5.**

> db.movies.find({$text:{$search:"gold dragon"}}).pretty()

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

**Delete Documents**

> db.movies.remove({title:"Pee Wee Herman's Big Adventure"})

WriteResult({ "nRemoved" : 1 })

> db.movies.remove({title:"Avatar"})

WriteResult({ "nRemoved" : 1 })

>

> db.movies.find().pretty()

{

"\_id" : ObjectId("6106cb951fb2405a3e0ef842"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : 1999,

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("6106cd991fb2405a3e0ef843"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : 1994,

"actors" : [

"Samuel L. Jackson",

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("6106ce621fb2405a3e0ef844"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : 2009,

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("6106cee21fb2405a3e0ef845"),

"title" : "The Hobbit:An Unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home- and the gold within it- from the dragon Smaug."

}

{

"\_id" : ObjectId("6106cf181fb2405a3e0ef846"),

"title" : "The Hobbit:The Desolation of Smaug",

"writer" : "J.R.R. Tolkein",

"year" : 2013,

"franchise" : "The Hobbit",

"synopsis" : "The dwarves, along with the Bilbo Baggins and Gandalf Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo baggins is in possession of a mysterious and magical ring."

}

{

"\_id" : ObjectId("6106d12d1fb2405a3e0ef848"),

"title" : "The Hobbit:The Battle of the Five Armies",

"writer" : "J.R.R. Tolkein",

"year" : 2012,

"franchise" : "The Hobbit",

"synopsis" : "Bilbo and Company are forced to enagage in a war against an array of combatants and keep the Lonely Mounatin from falling into the hands of the rising darkness"

}

**Relationships**

**Insert users collection**

> db.users.insert({username:"GoodGuyGreg",first\_name:"Good Guy",last\_name:"Greg"})

WriteResult({ "nInserted" : 1 })

> db.users.insert({username:"ScumBagSteve",full\_name:{first:"Scumbag",last:"steve"}})

WriteResult({ "nInserted" : 1 })

**Insert posts collection**

> db.posts.insert([{username:"GoodGuyGreg",title:"Passes out at party",body:"wakes up early and cleans house"},{username:"GoodGuyGreg",title:"Steals your identity",body:"Raises your credit score"},{username:"GoodGuyGreg",title:"Reports a bug in your code",body:"Sends you a pull Request"},{username:"ScumbagSteve",title:"Borrows something",body:"Sells it"},{username:"ScumbagSteve",title:"Borrows everything",body:"The end"},{username:"ScumbagSteve",title:"forks your repo on github",body:"Sets to private"}])

BulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 6,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

})

**Insert comments collection**

> db.comments.insert({username:"GoodGuyGreg",comment:"Hope you got a good deal!",post:ObjectId("6107bc725afbeb8c8e05725e")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"GoodGuyGreg",comment:"What's mine is yours!",post:ObjectId("6107bc725afbeb8c8e05725f")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"GoodGuyGreg",comment:"Don't violate the licensing agreement!",post:ObjectId("6107bc725afbeb8c8e057260")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"ScumbagSteve",comment:"It still isn't clean",post:ObjectId("6107bc725afbeb8c8e05725b")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"ScumbagSteve",comment:"Denied your PR cause I found a hack",post:ObjectId("6107bc725afbeb8c8e05725d")})

WriteResult({ "nInserted" : 1 })

**Querying related collections**

**1.**

> db.users.find().pretty()

{

"\_id" : ObjectId("6107b70c5afbeb8c8e057257"),

"username" : "GoodGuyGreg",

"first\_name" : "Good Guy",

"last\_name" : "Greg"

}

{

"\_id" : ObjectId("6107b7355afbeb8c8e057258"),

"username" : "ScumBagSteve",

"full\_name" : {

"first" : "Scumbag",

"last" : "steve"

}

}

**2.**

> db.posts.find().pretty()

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725b"),

"username" : "GoodGuyGreg",

"title" : "Passes out at party",

"body" : "wakes up early and cleans house"

}

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725c"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725d"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a pull Request"

}

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725e"),

"username" : "ScumbagSteve",

"title" : "Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725f"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

"\_id" : ObjectId("6107bc725afbeb8c8e057260"),

"username" : "ScumbagSteve",

"title" : "forks your repo on github",

"body" : "Sets to private"

}

**3.**

> db.posts.find({username:"GoodGuyGreg"})

{ "\_id" : ObjectId("6107bc725afbeb8c8e05725b"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "wakes up early and cleans house" }

{ "\_id" : ObjectId("6107bc725afbeb8c8e05725c"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("6107bc725afbeb8c8e05725d"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull Request" }

**4.**

> db.posts.find({username:"ScumbagSteve"})

{ "\_id" : ObjectId("6107bc725afbeb8c8e05725e"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("6107bc725afbeb8c8e05725f"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("6107bc725afbeb8c8e057260"), "username" : "ScumbagSteve", "title" : "forks your repo on github", "body" : "Sets to private" }

>

**5.**

> db.comments.find().pretty()

{

"\_id" : ObjectId("6107c0f75afbeb8c8e05726d"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("6107bc725afbeb8c8e05725e")

}

{

"\_id" : ObjectId("6107c1cc5afbeb8c8e05726e"),

"username" : "GoodGuyGreg",

"comment" : "What's mine is yours!",

"post" : ObjectId("6107bc725afbeb8c8e05725f")

}

{

"\_id" : ObjectId("6107c2425afbeb8c8e05726f"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("6107bc725afbeb8c8e057260")

}

{

"\_id" : ObjectId("6107c2ab5afbeb8c8e057270"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("6107bc725afbeb8c8e05725b")

}

{

"\_id" : ObjectId("6107c3255afbeb8c8e057271"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("6107bc725afbeb8c8e05725d")

}

**6.**

> db.comments.find({username:"GoodGuyGreg"})

{ "\_id" : ObjectId("6107c0f75afbeb8c8e05726d"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("6107bc725afbeb8c8e05725e") }

{ "\_id" : ObjectId("6107c1cc5afbeb8c8e05726e"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("6107bc725afbeb8c8e05725f") }

{ "\_id" : ObjectId("6107c2425afbeb8c8e05726f"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("6107bc725afbeb8c8e057260") }

>

**7.**

> db.comments.find({username:"ScumbagSteve"}).pretty()

{

"\_id" : ObjectId("6107c2ab5afbeb8c8e057270"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("6107bc725afbeb8c8e05725b")

}

{

"\_id" : ObjectId("6107c3255afbeb8c8e057271"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("6107bc725afbeb8c8e05725d")

}

**8.**

> db.posts.find({title:"Reports a bug in your code"}).pretty()

{

"\_id" : ObjectId("6107bc725afbeb8c8e05725d"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a pull Request"

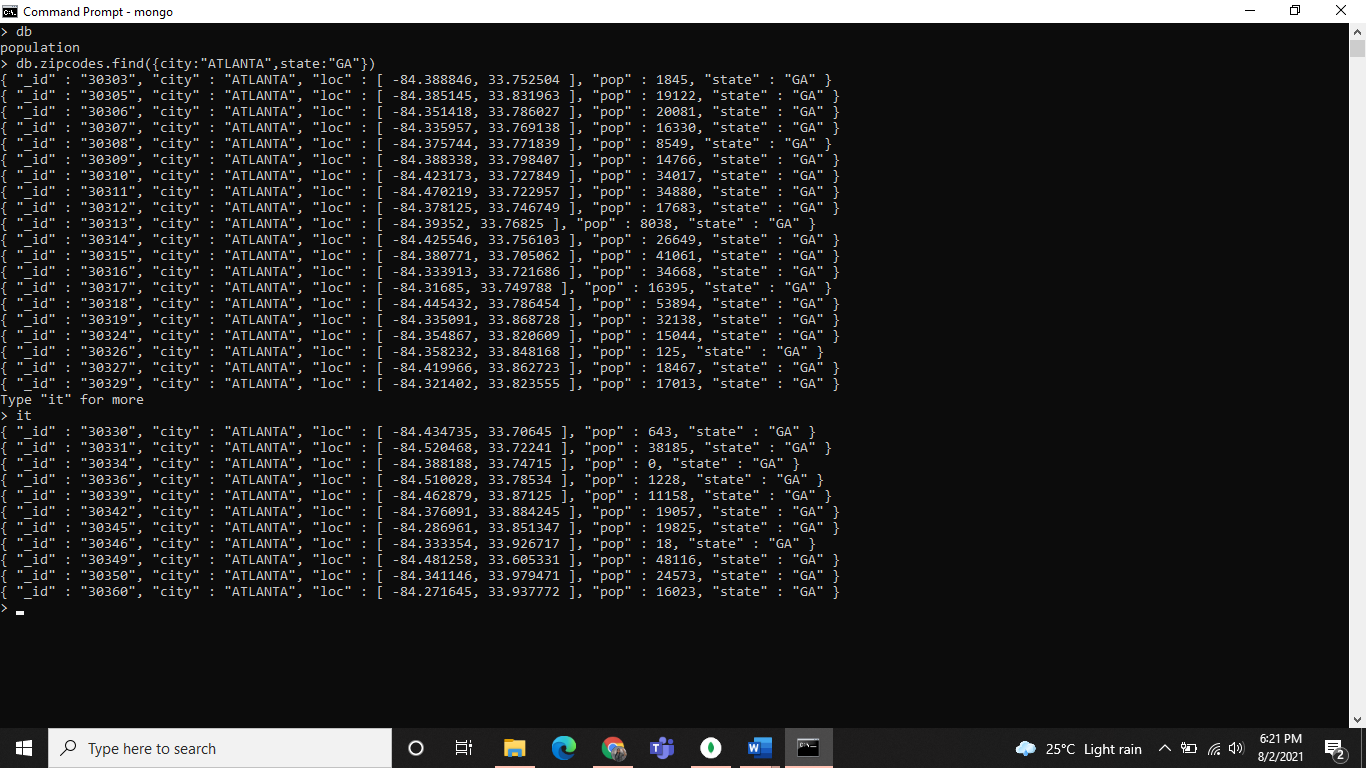
}

------------------------------------------------------------------------------------------------------------------------------------------

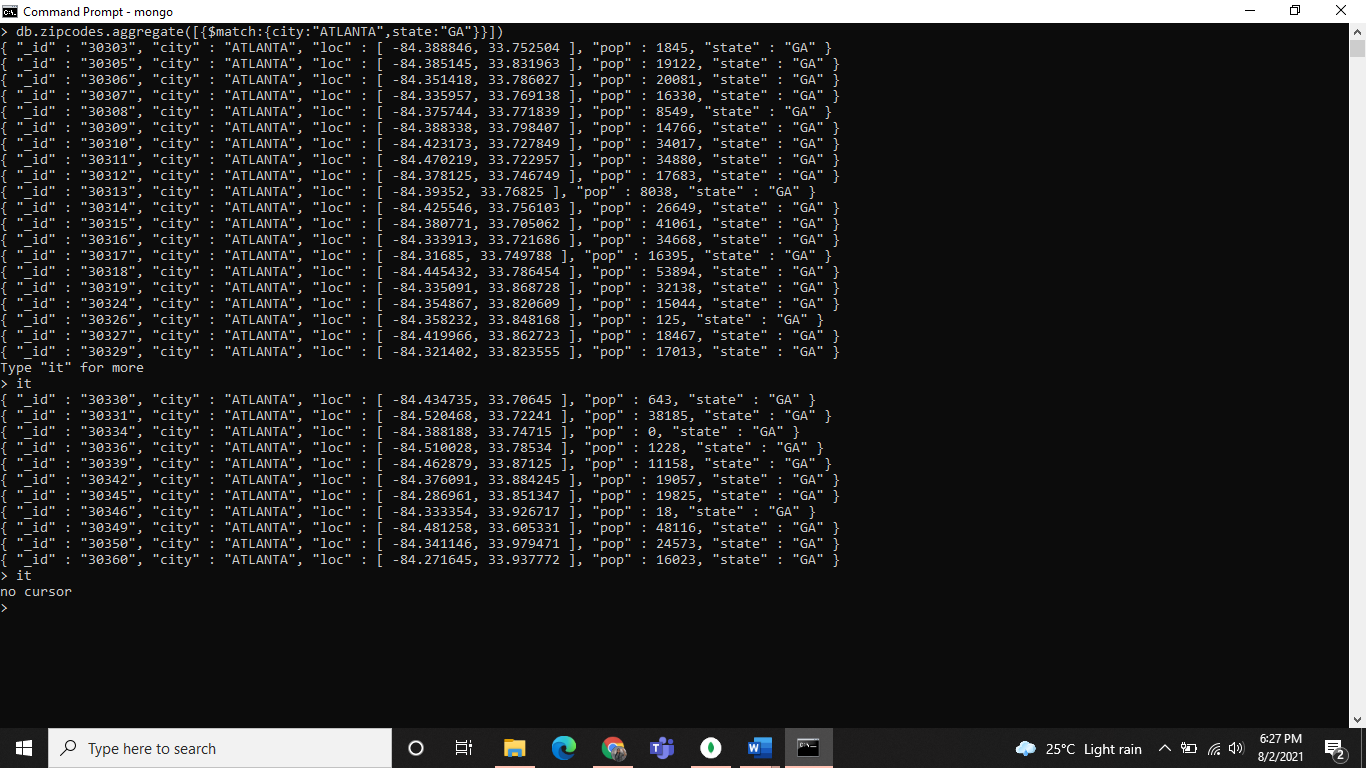
**Assignment 2**

**Atlanta Population:**

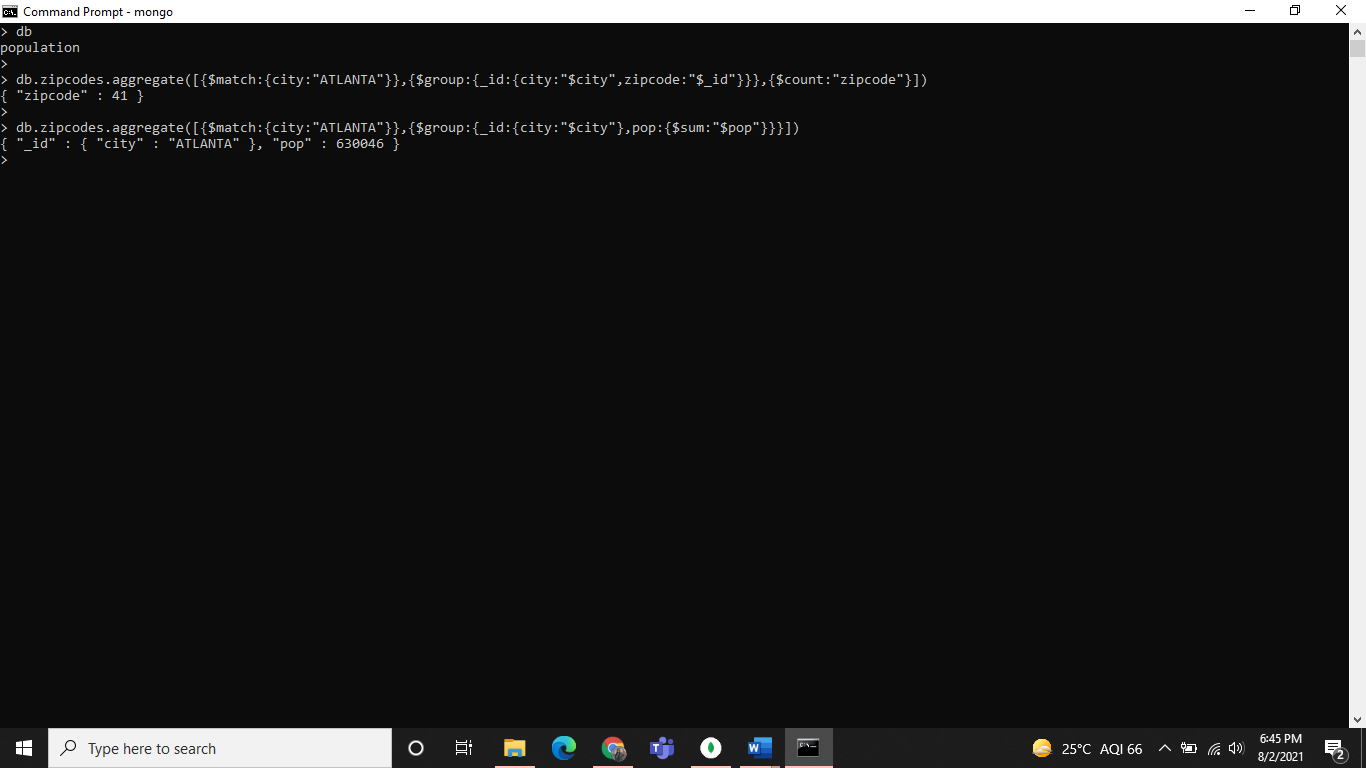
Q.1



Q.2

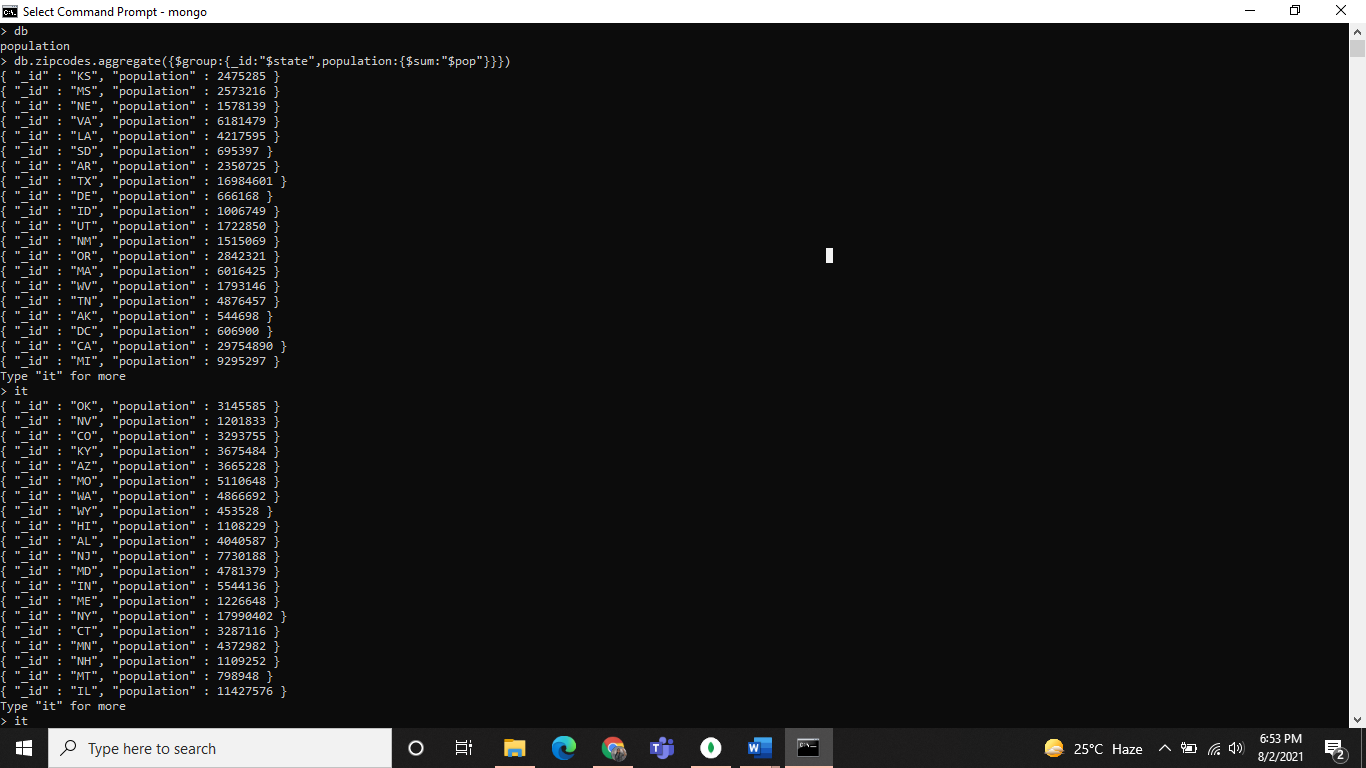


Q.3 & Q.4



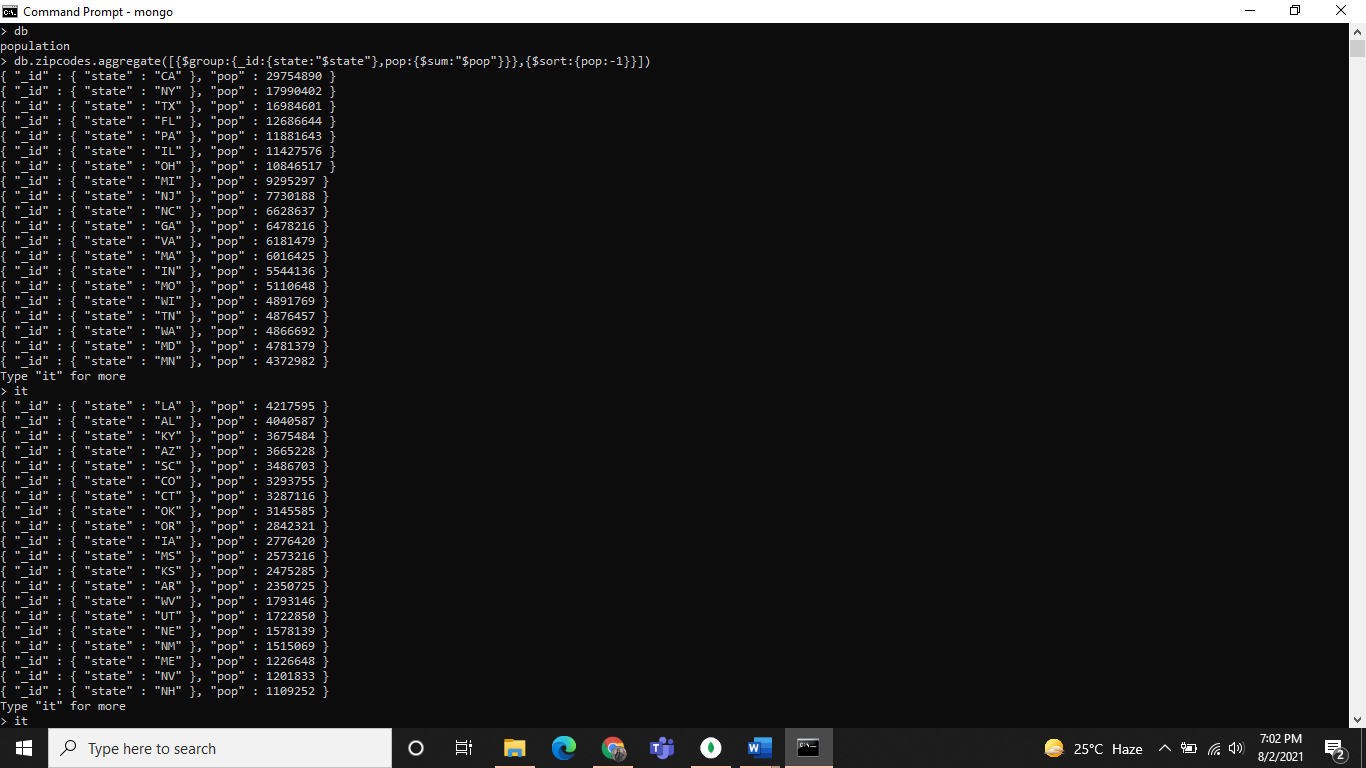
**Populations by State:**

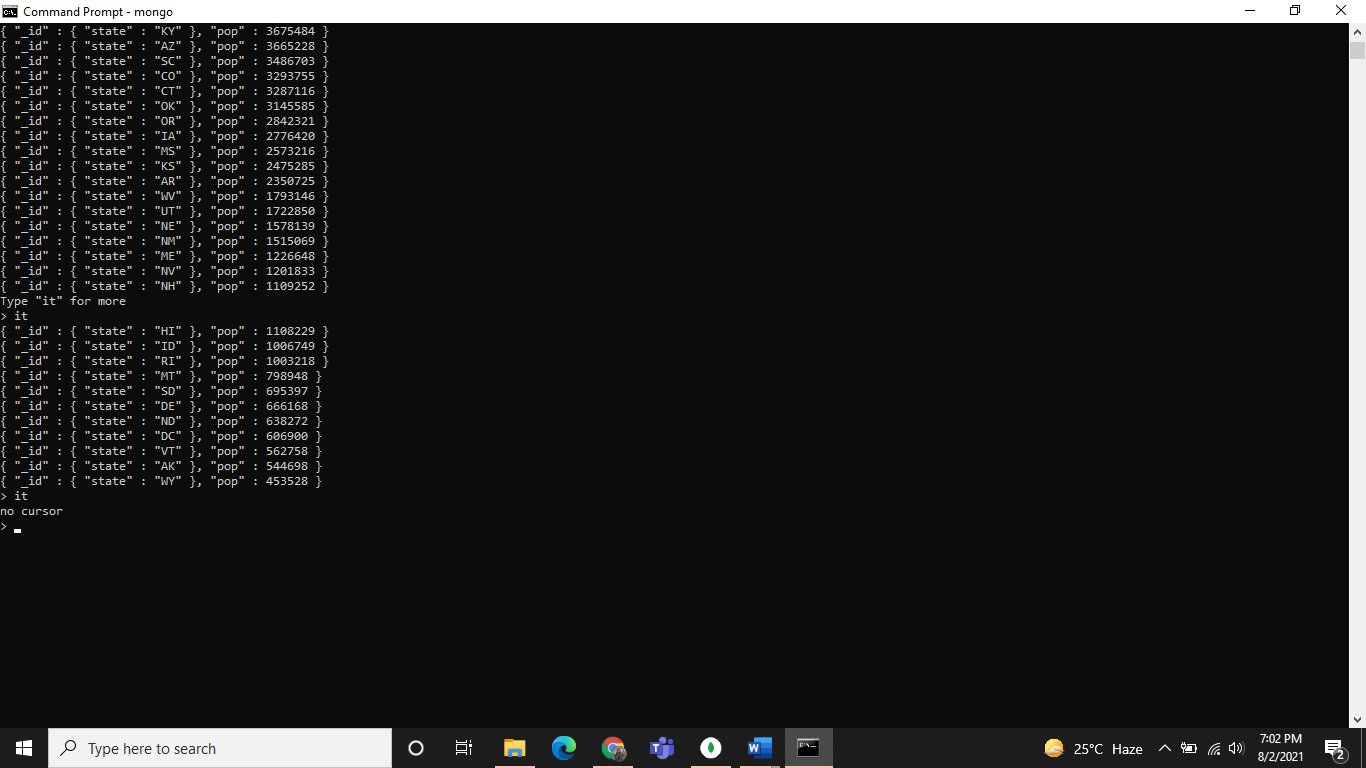
Q.1



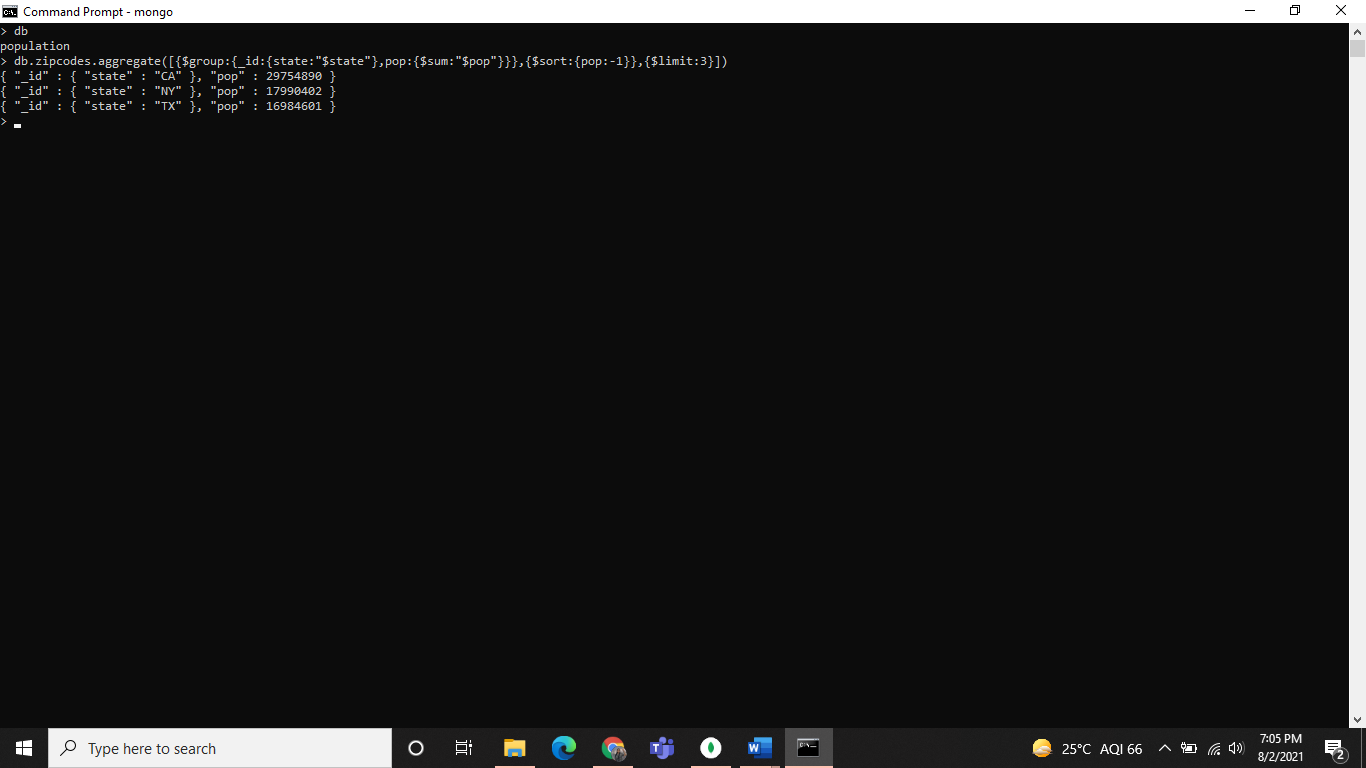


Q.2





Q.3

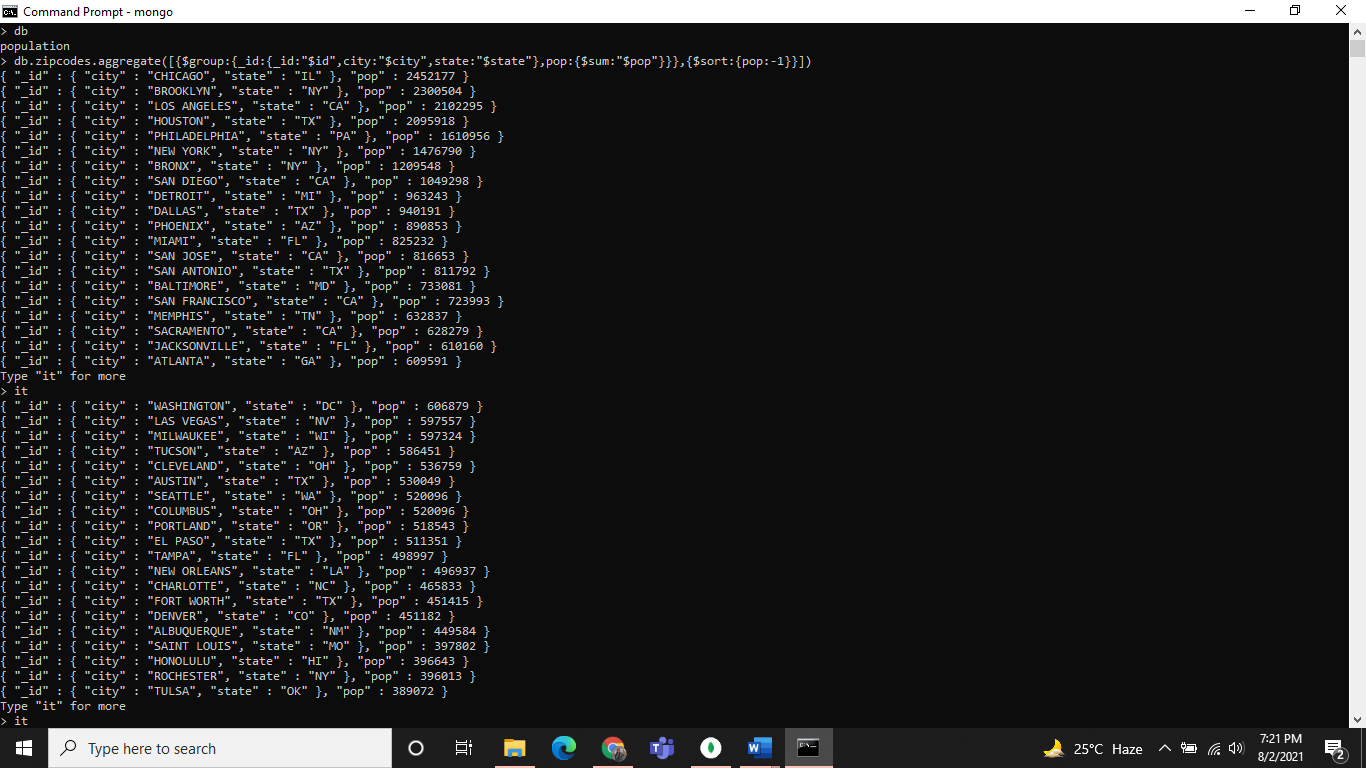


**Populations by city:**

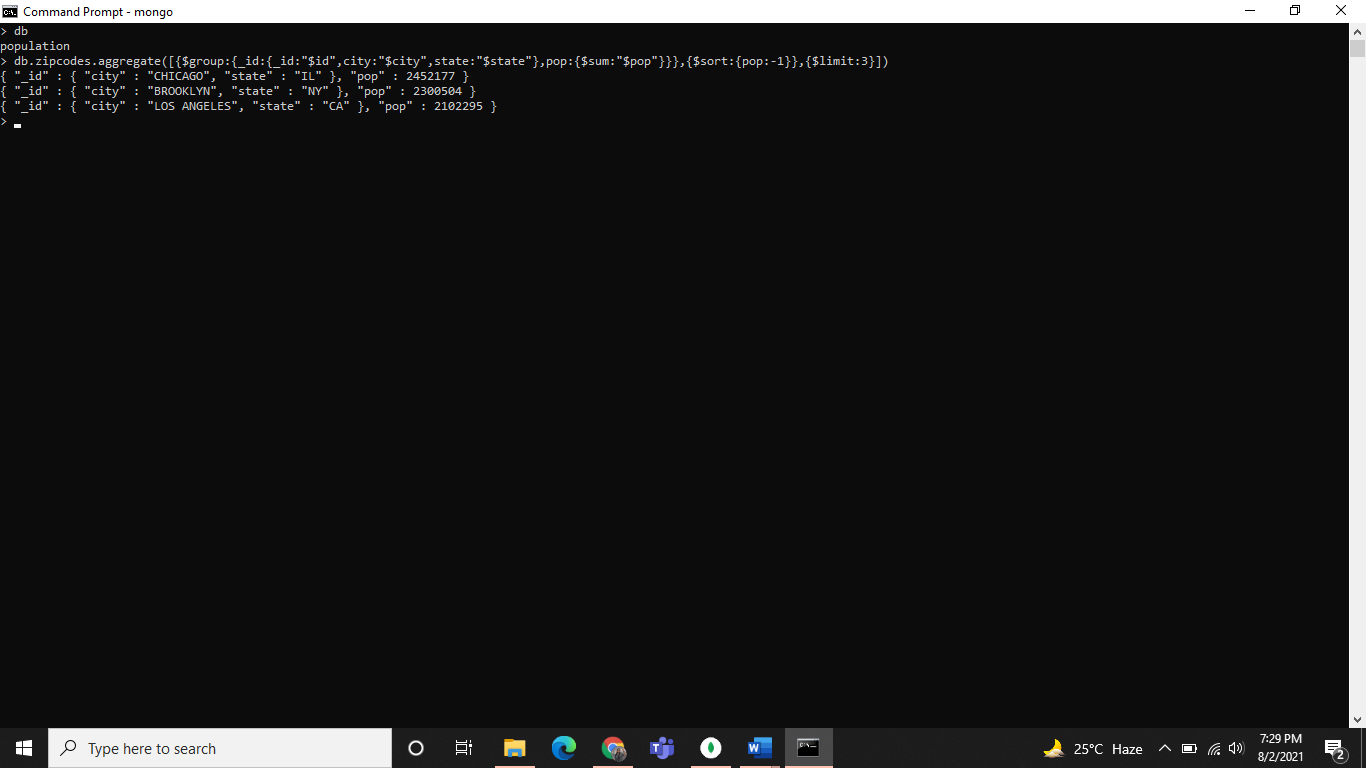
Q.1

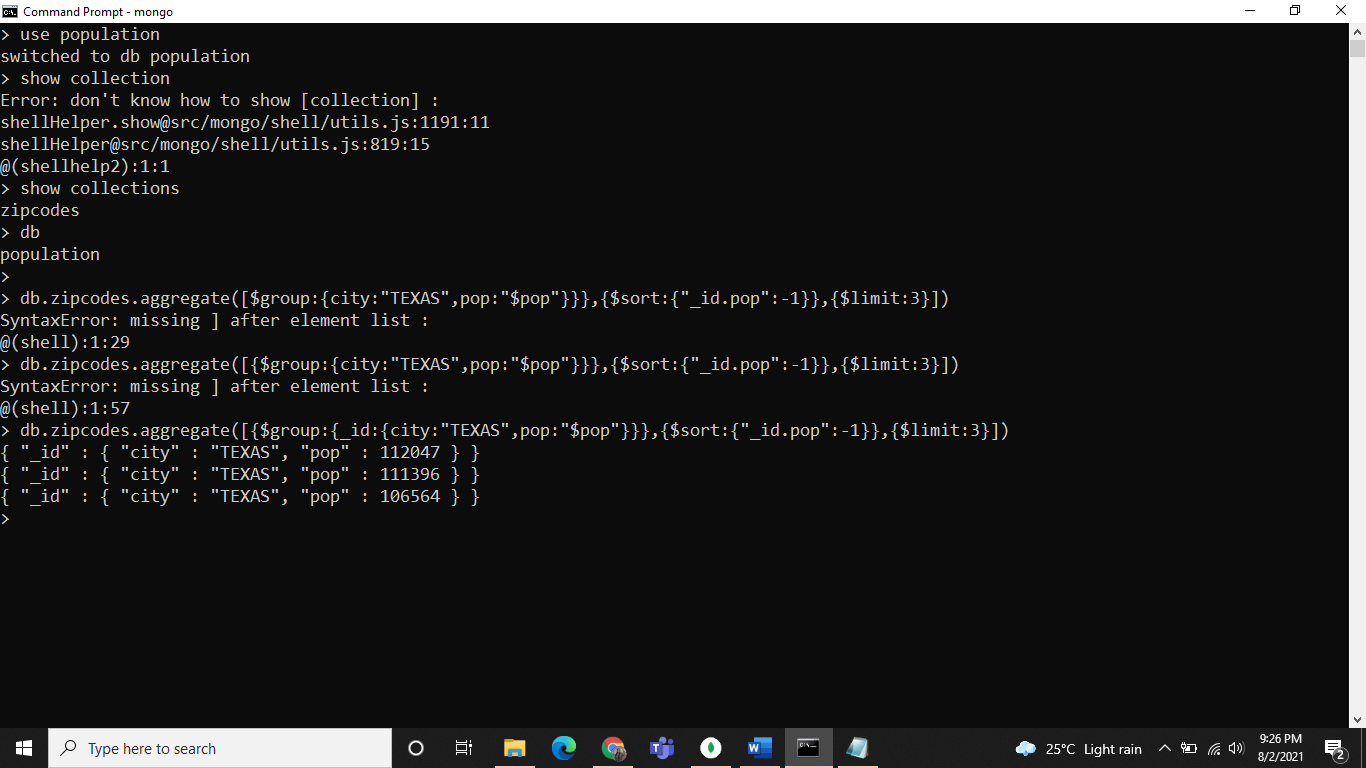


Q.2



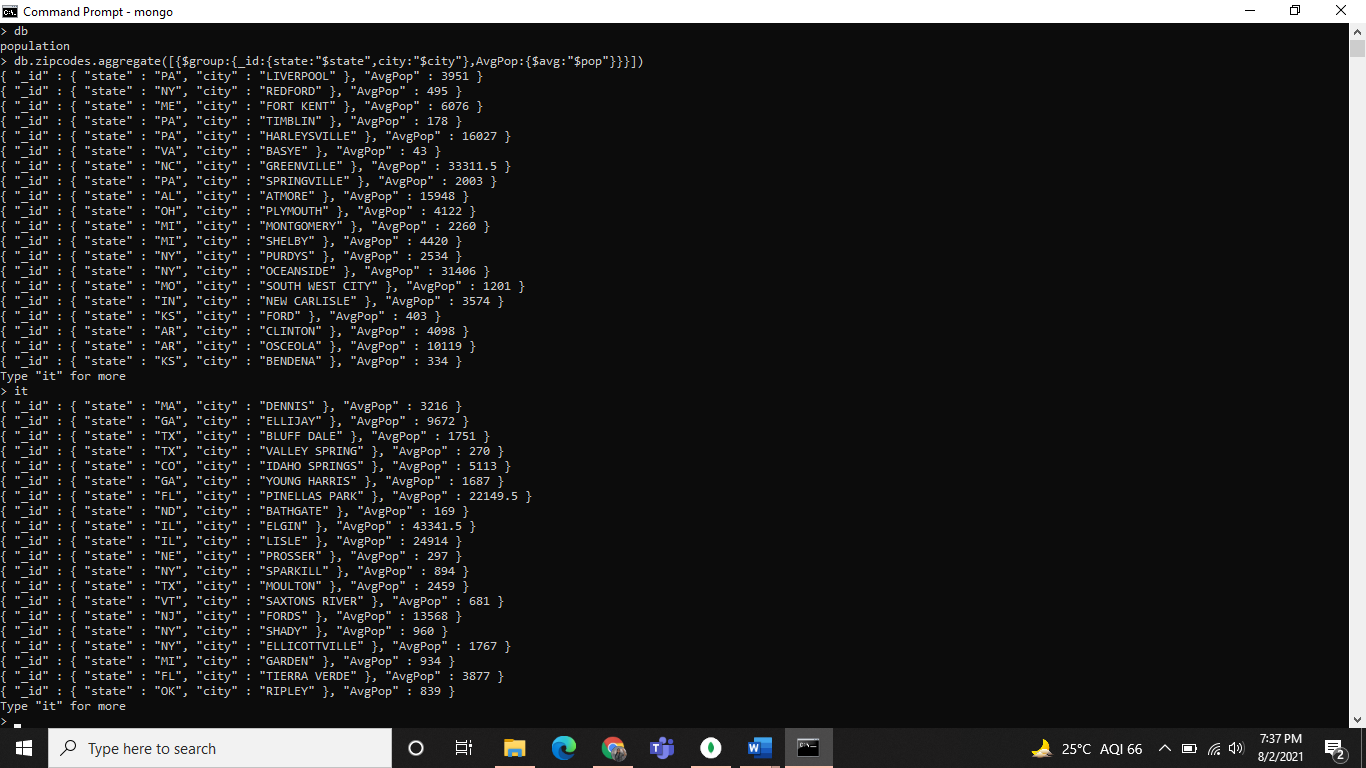
Q.3



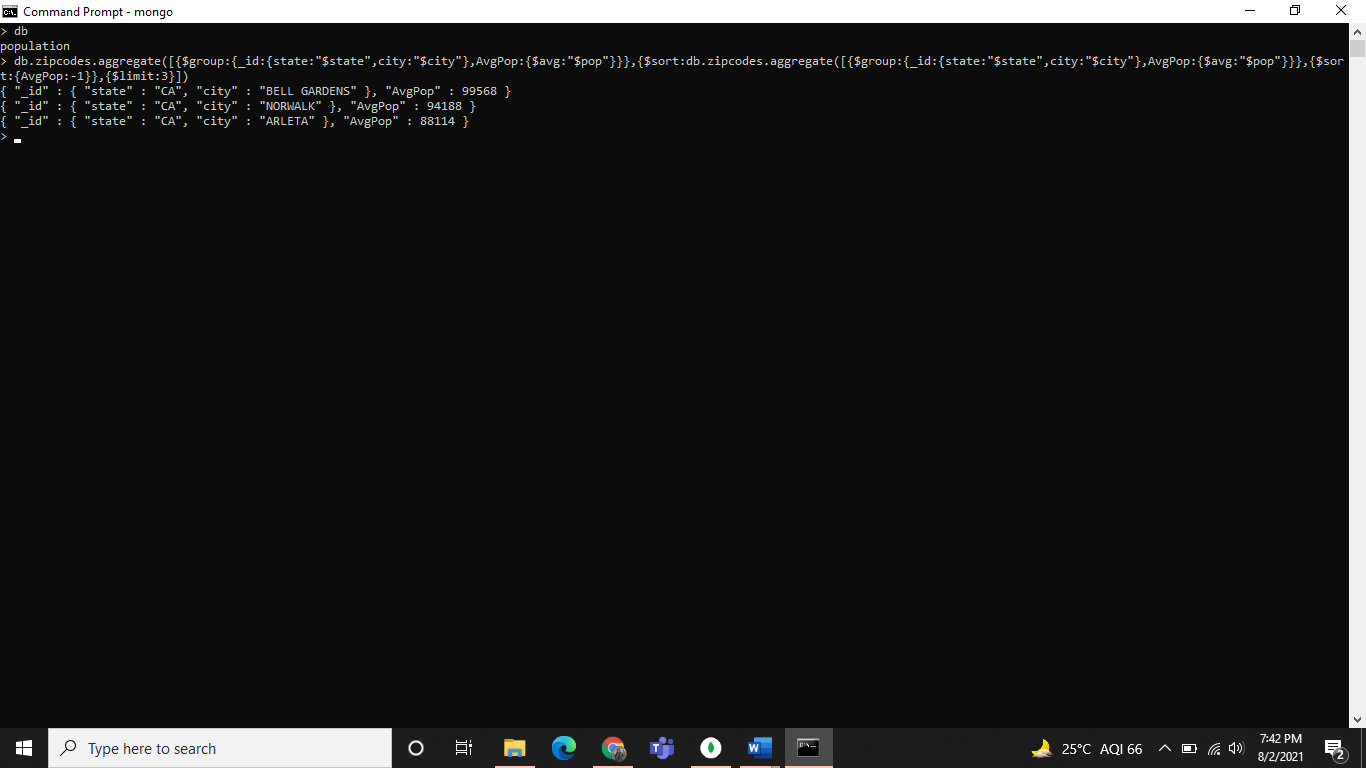
Q.4 

**Bonus:**

Q.1



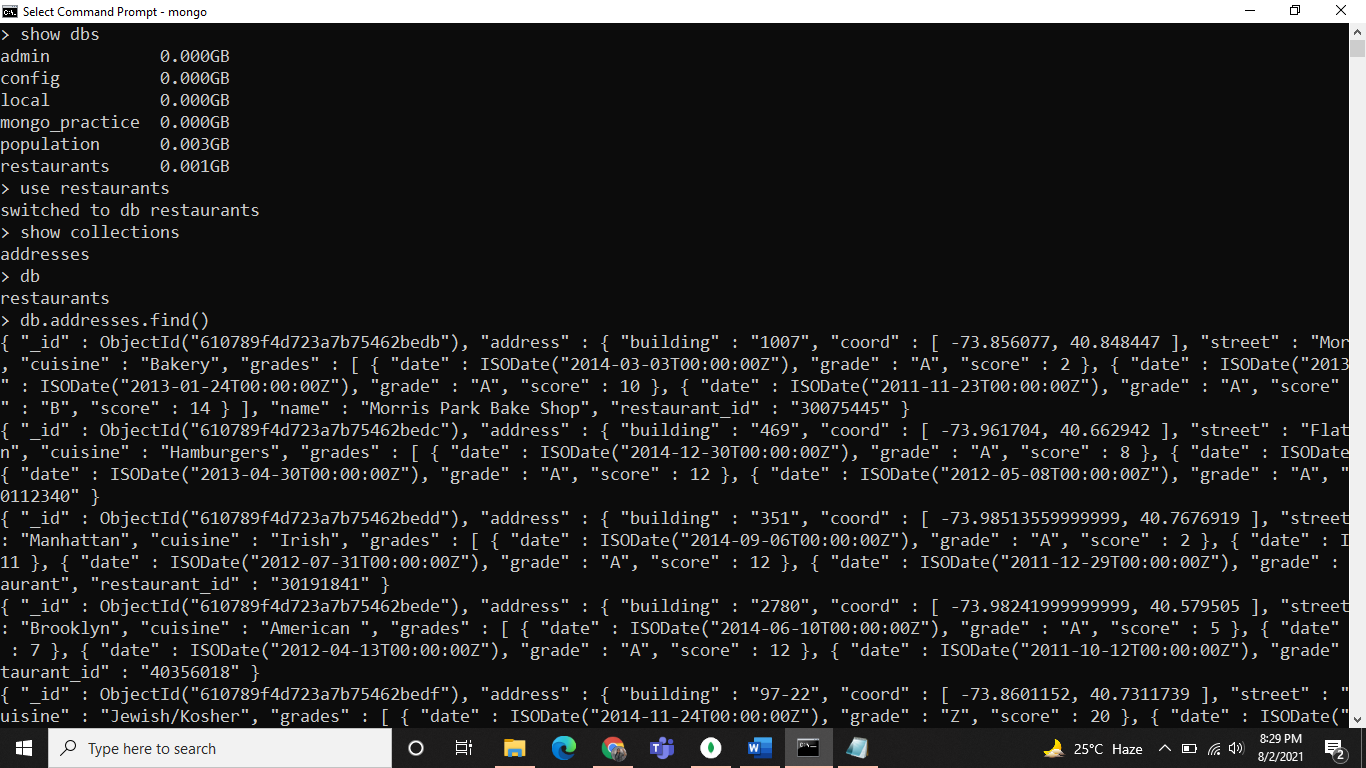
Q.2



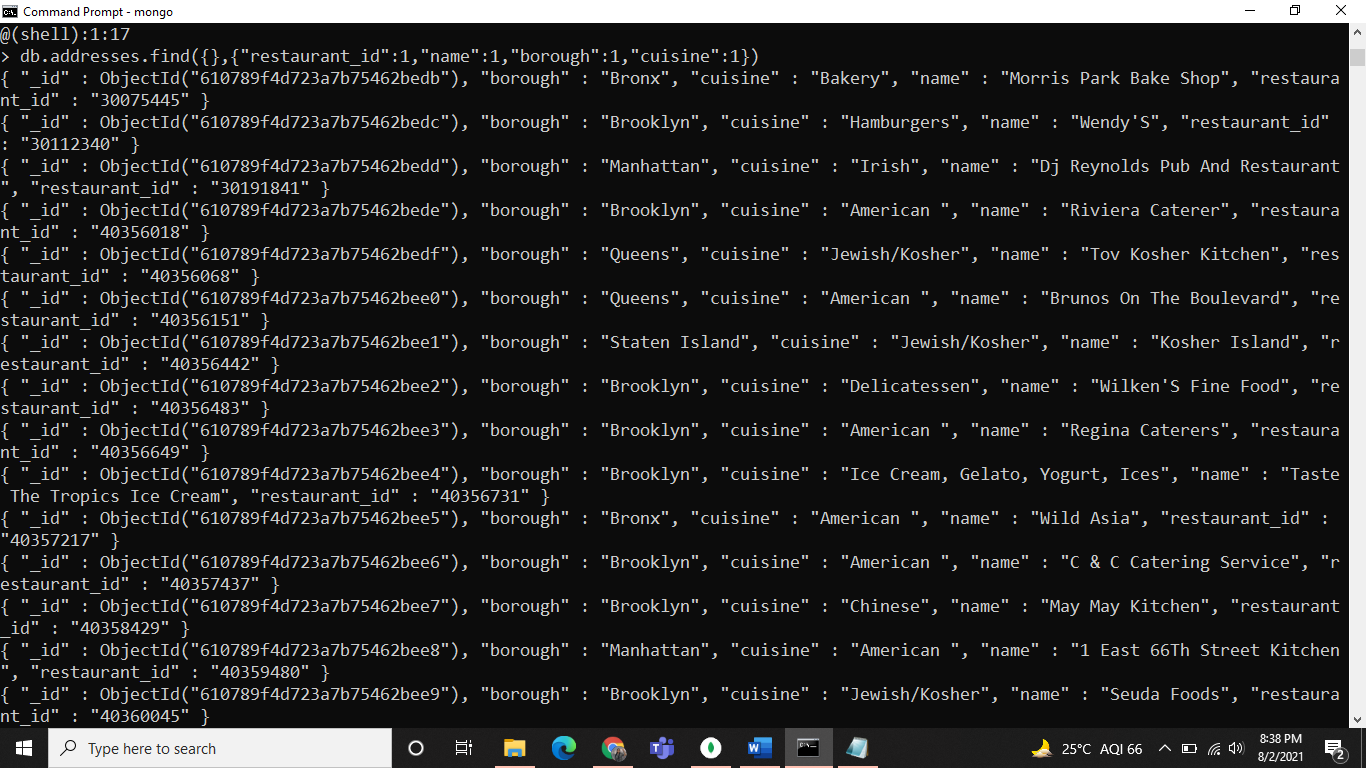
**------------------------------------------------------------------------------------**

**Assignment 3**

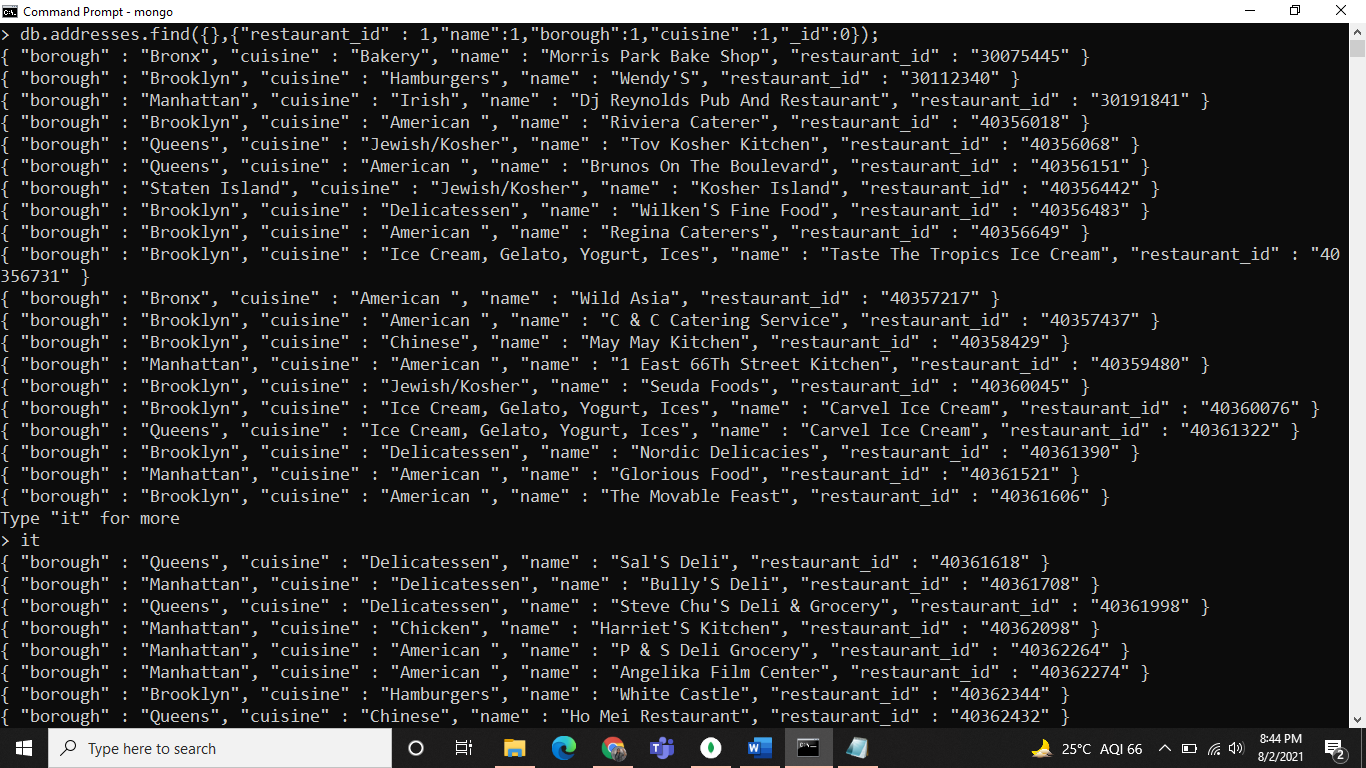
Q.1



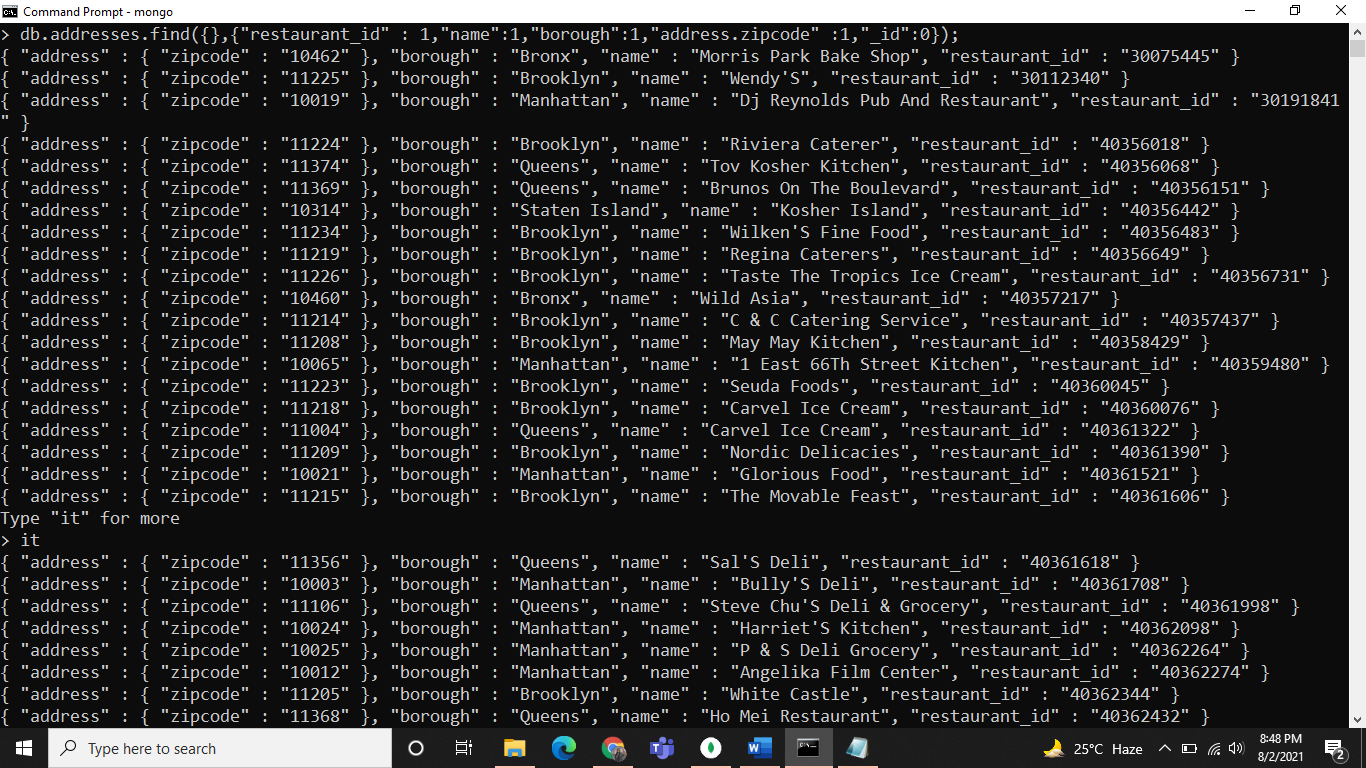
Q.2



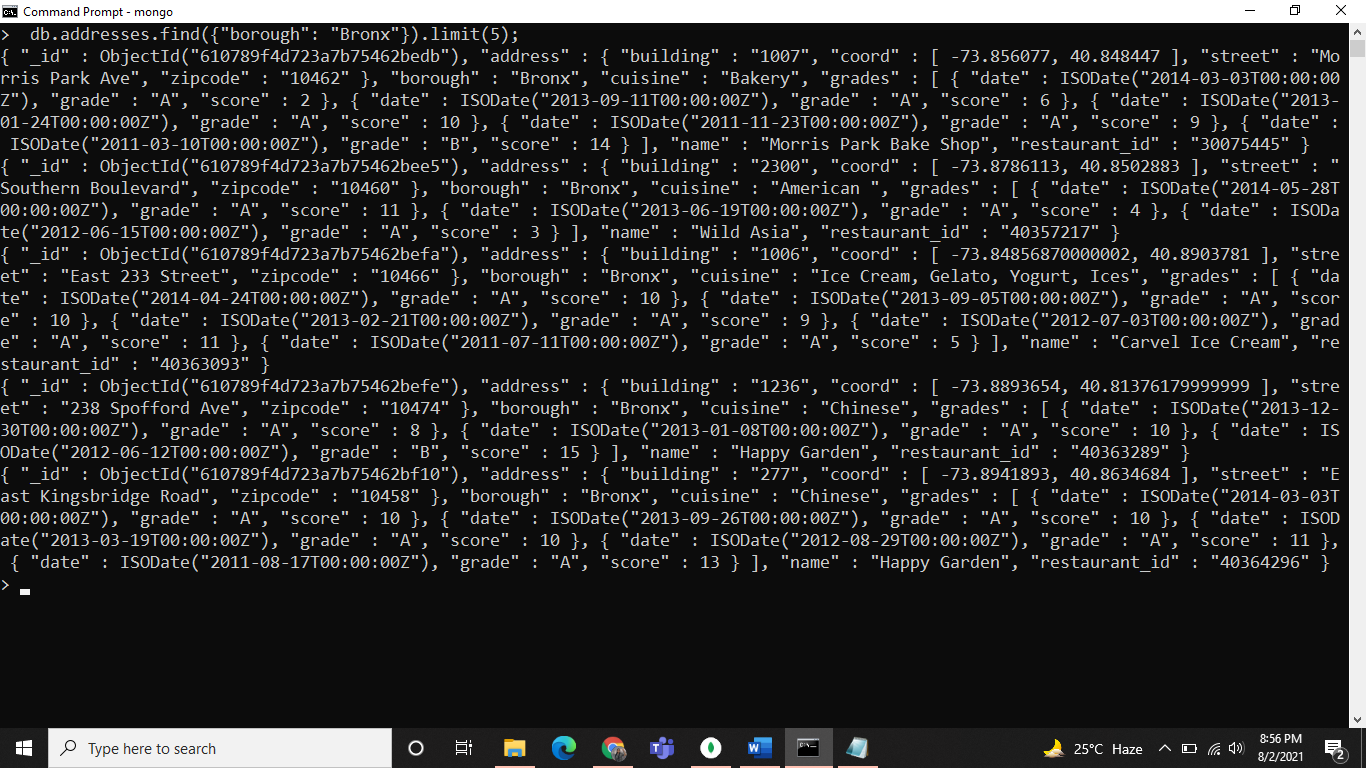
Q.3



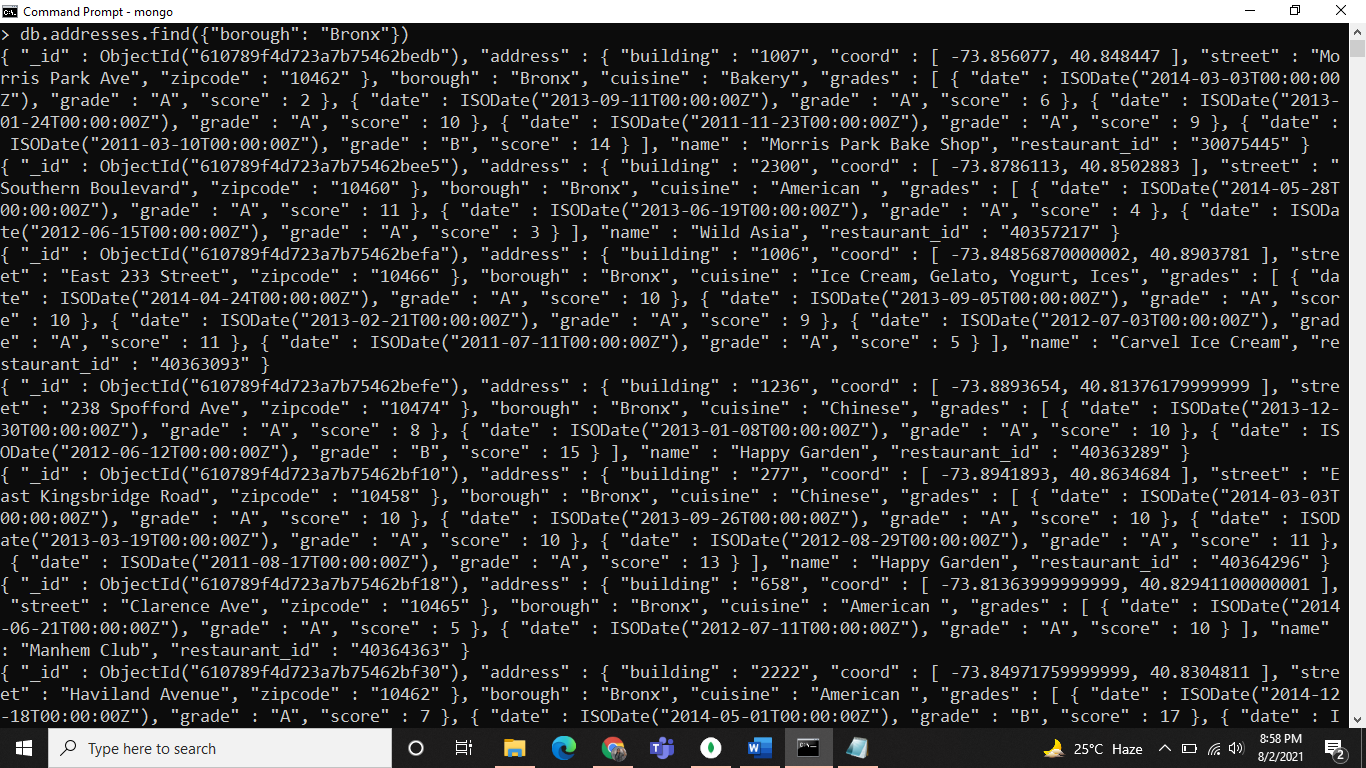
Q.4



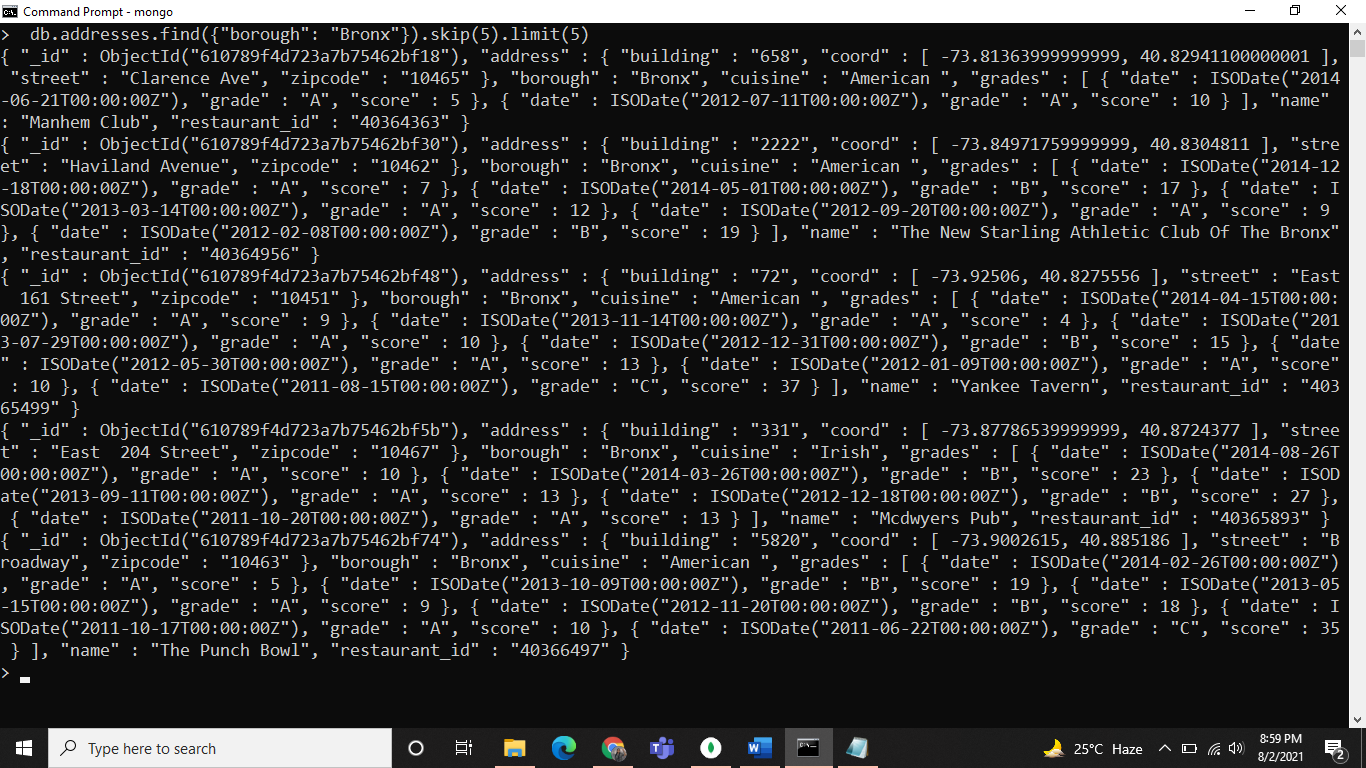
Q.5



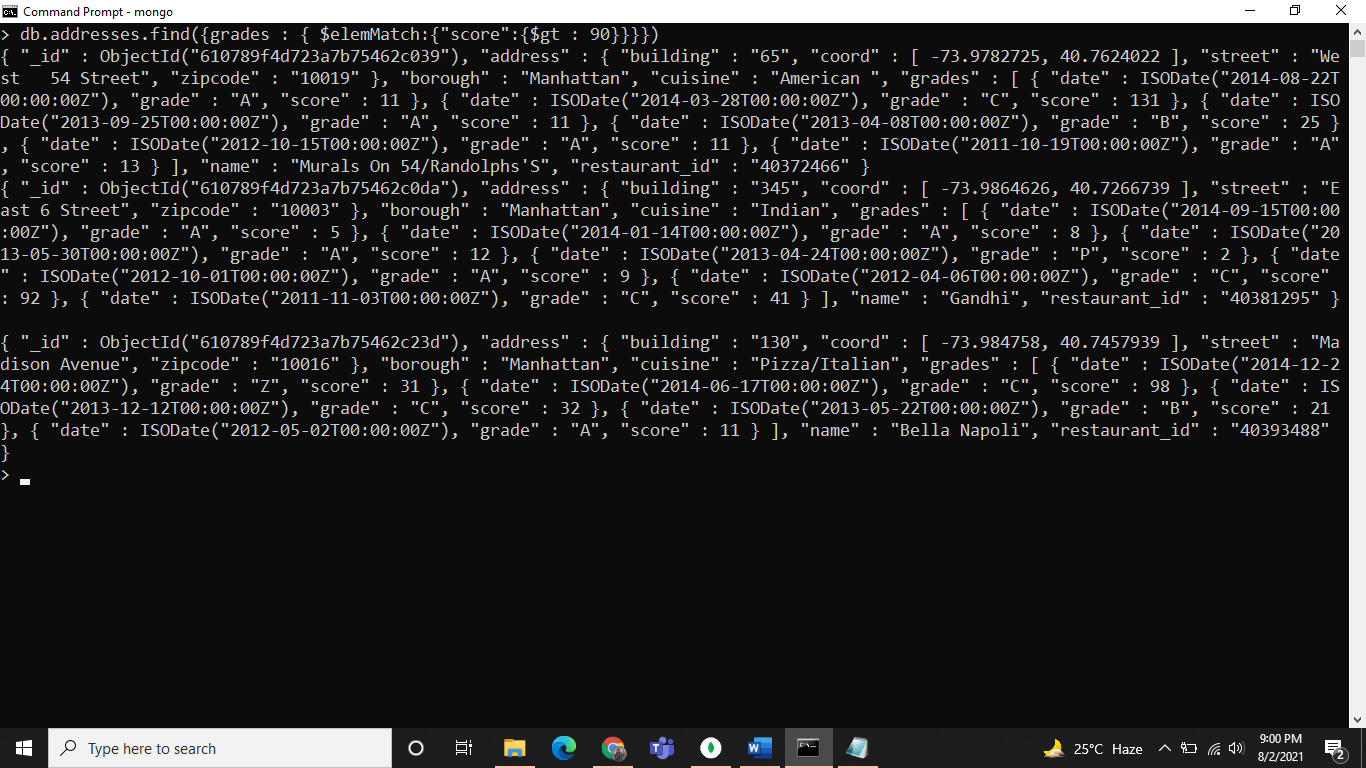
Q.6



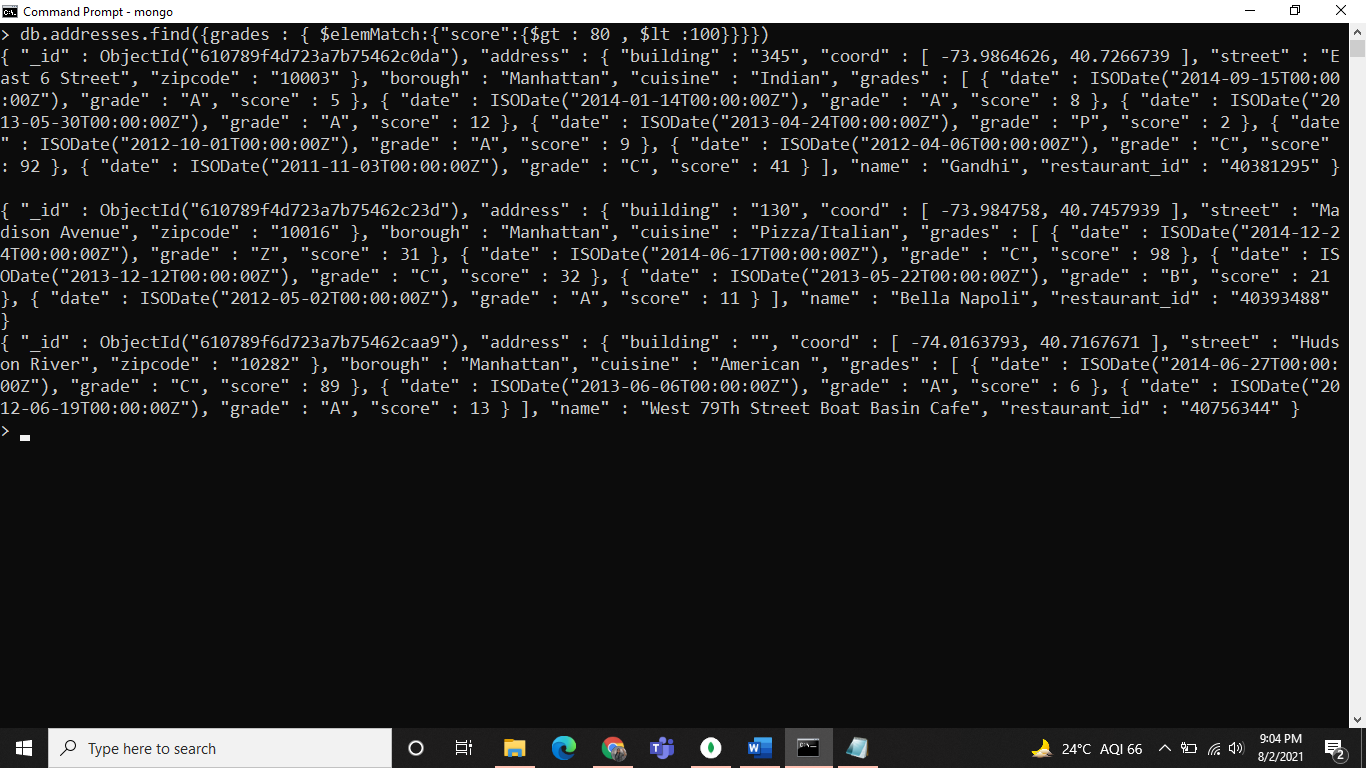
Q.7



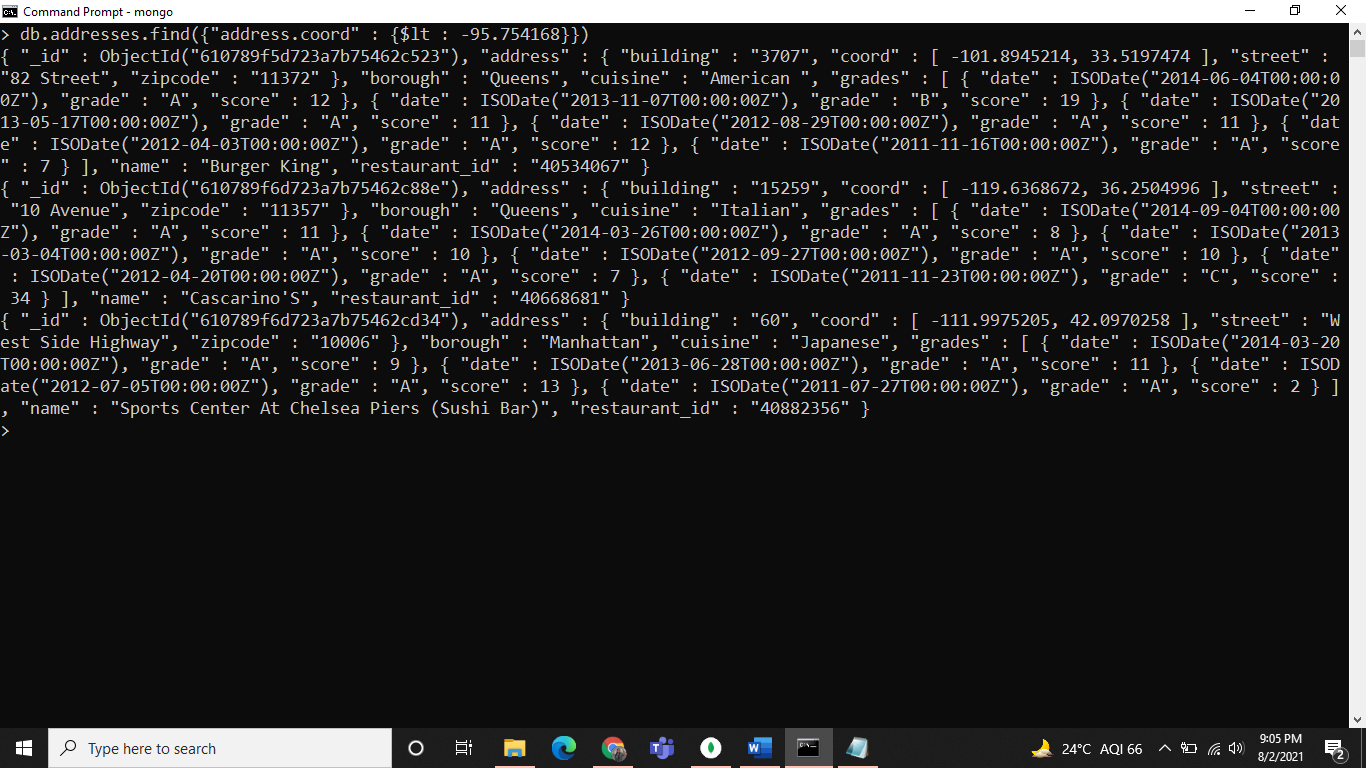
Q.8



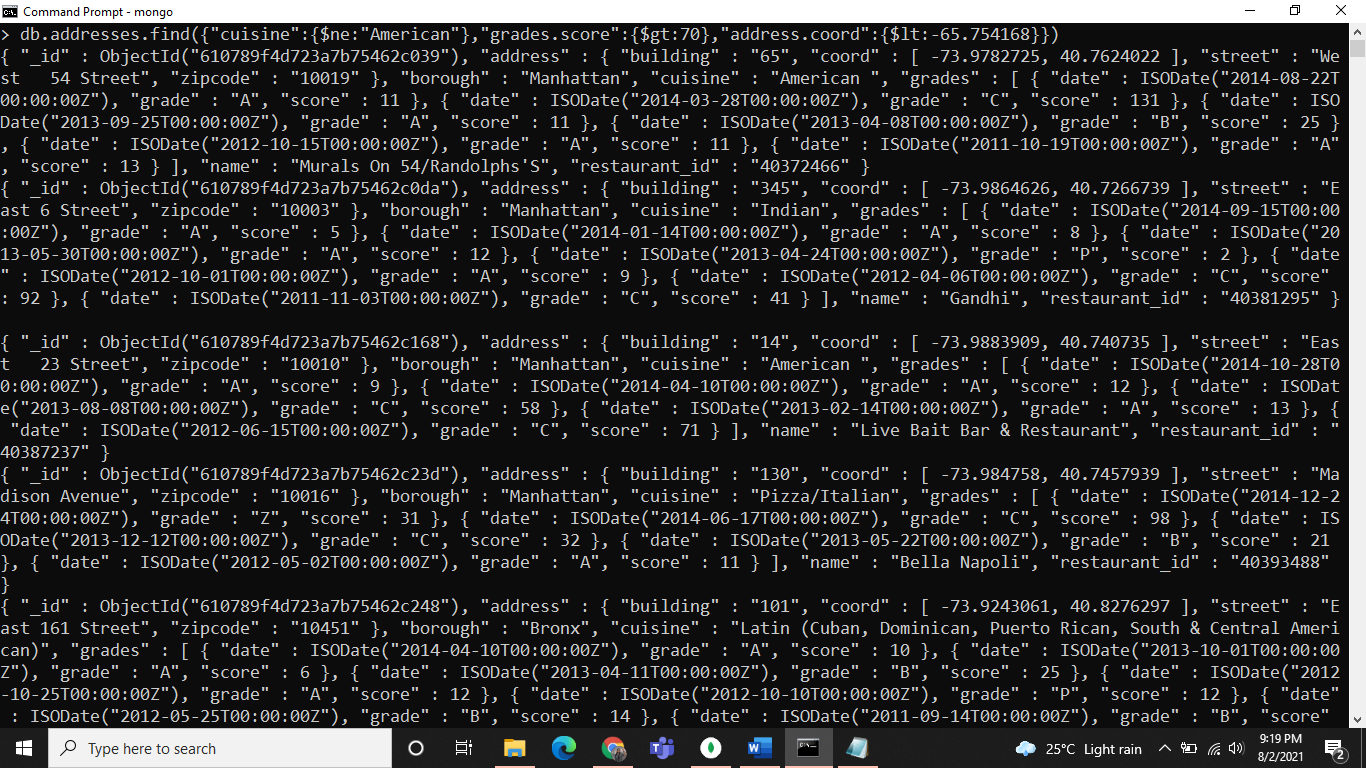
Q.9



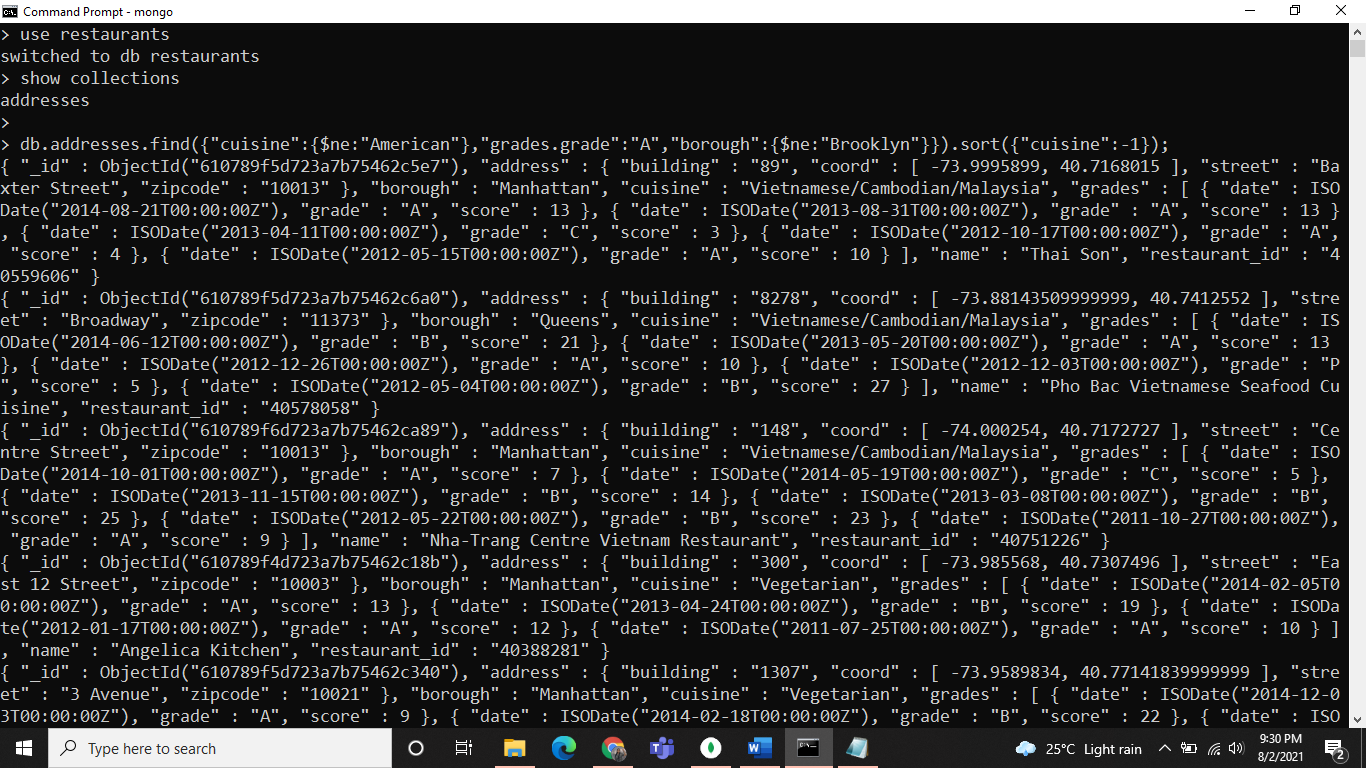
Q.10



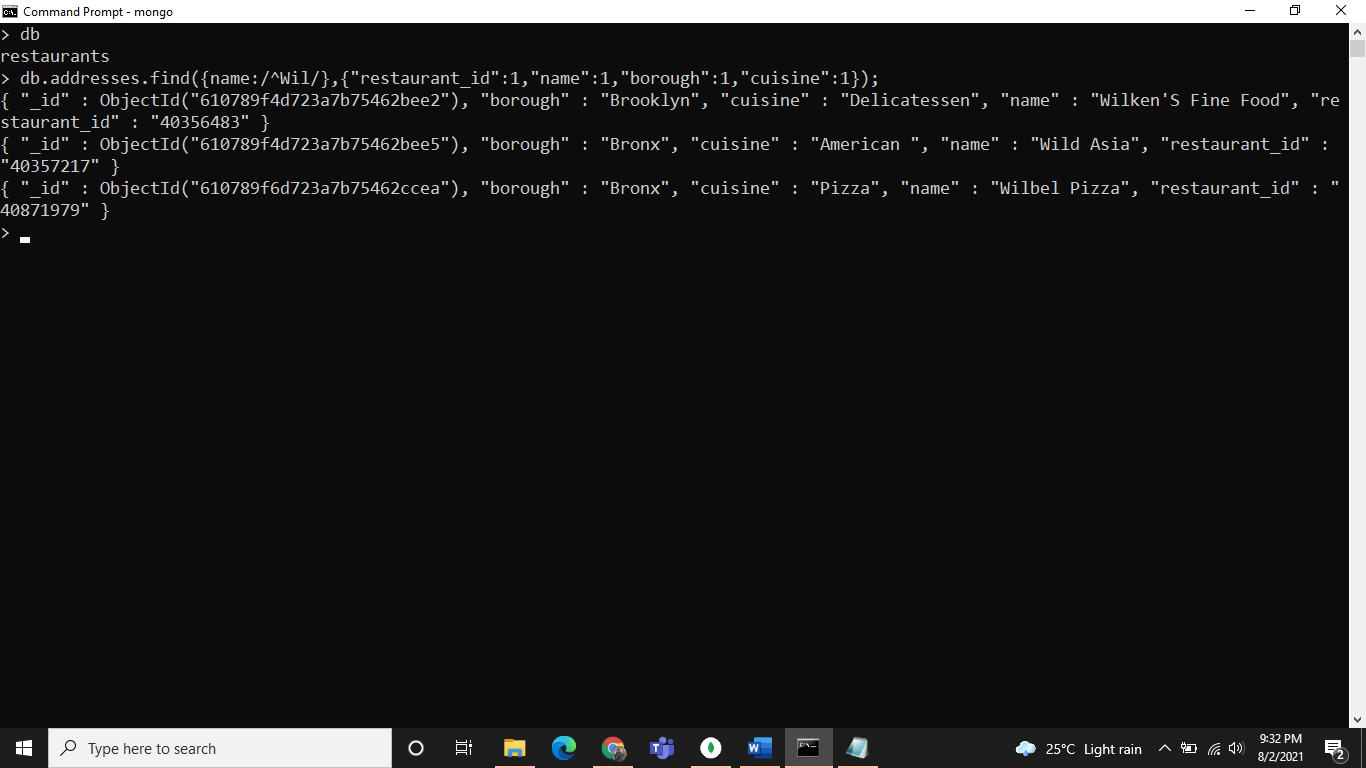
Q.11, 12



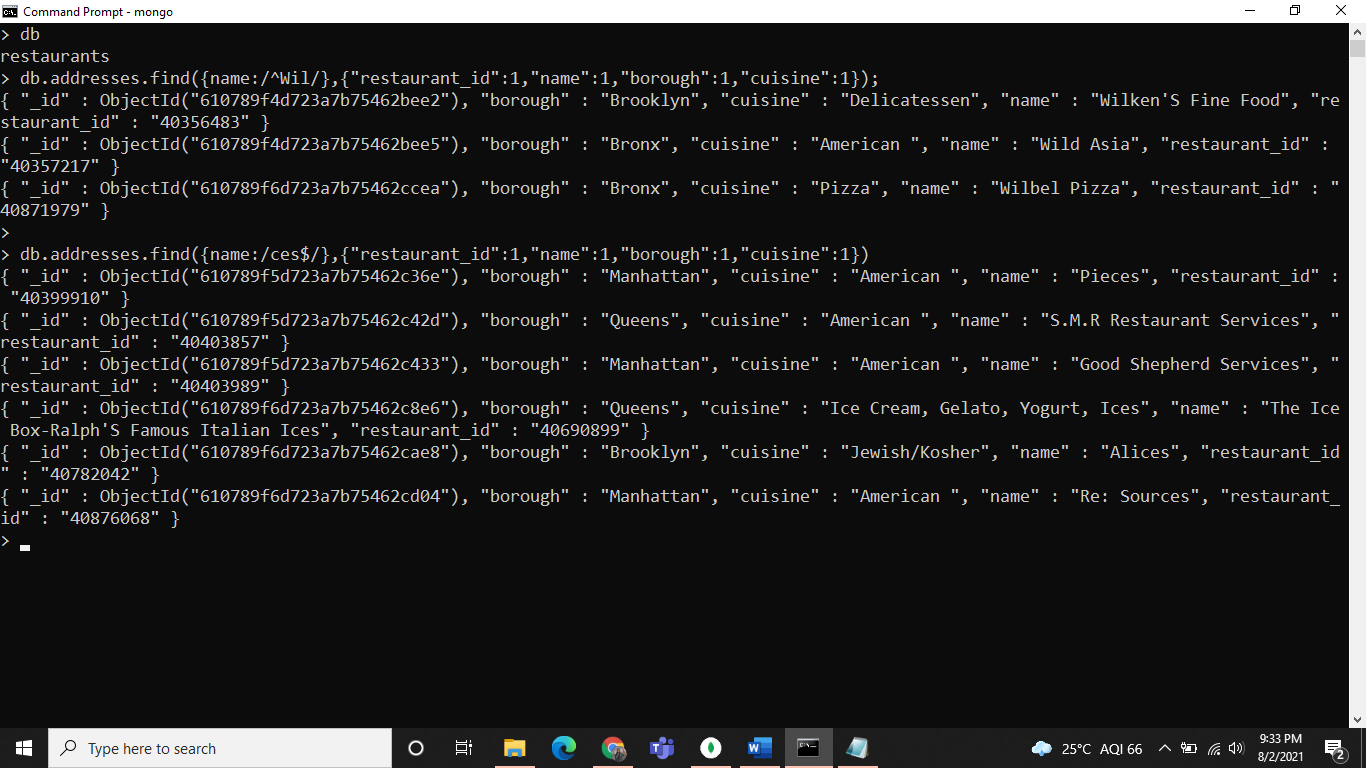
Q.13



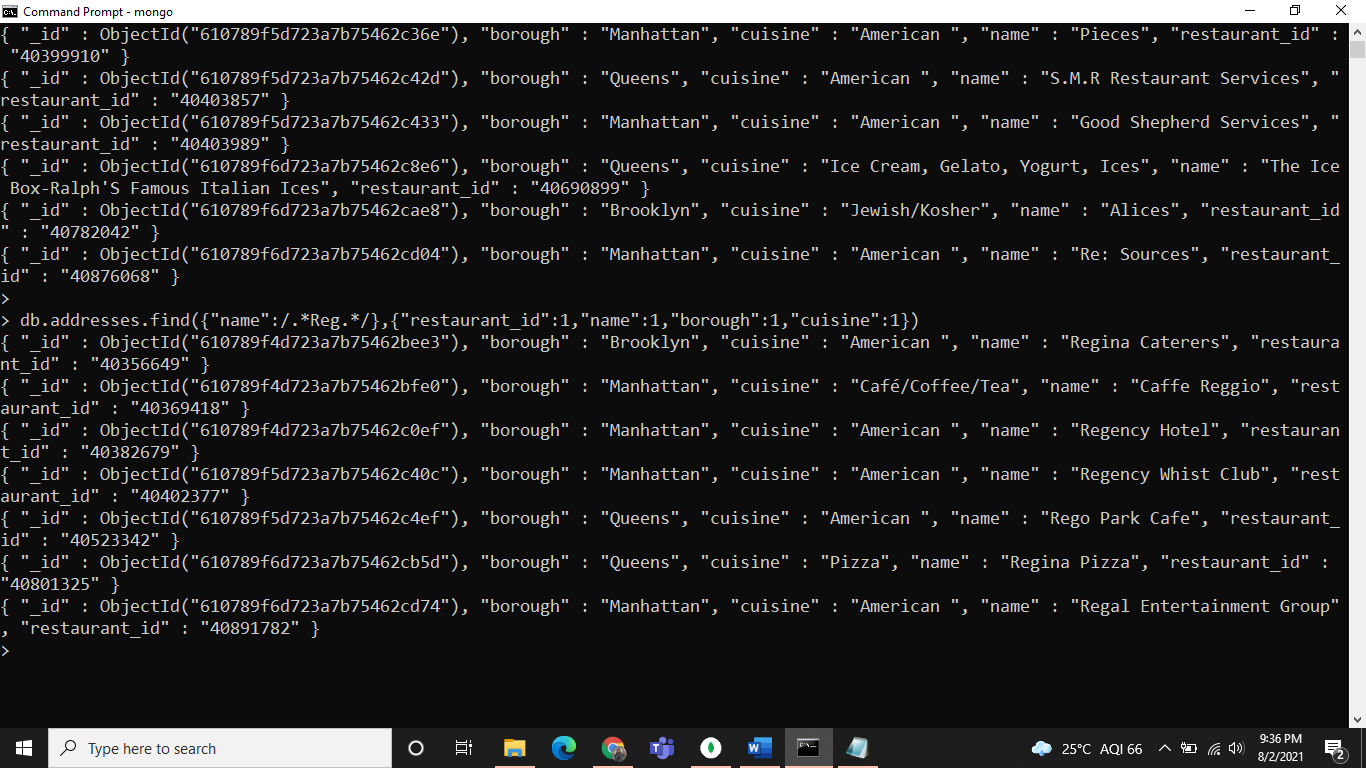
Q.14



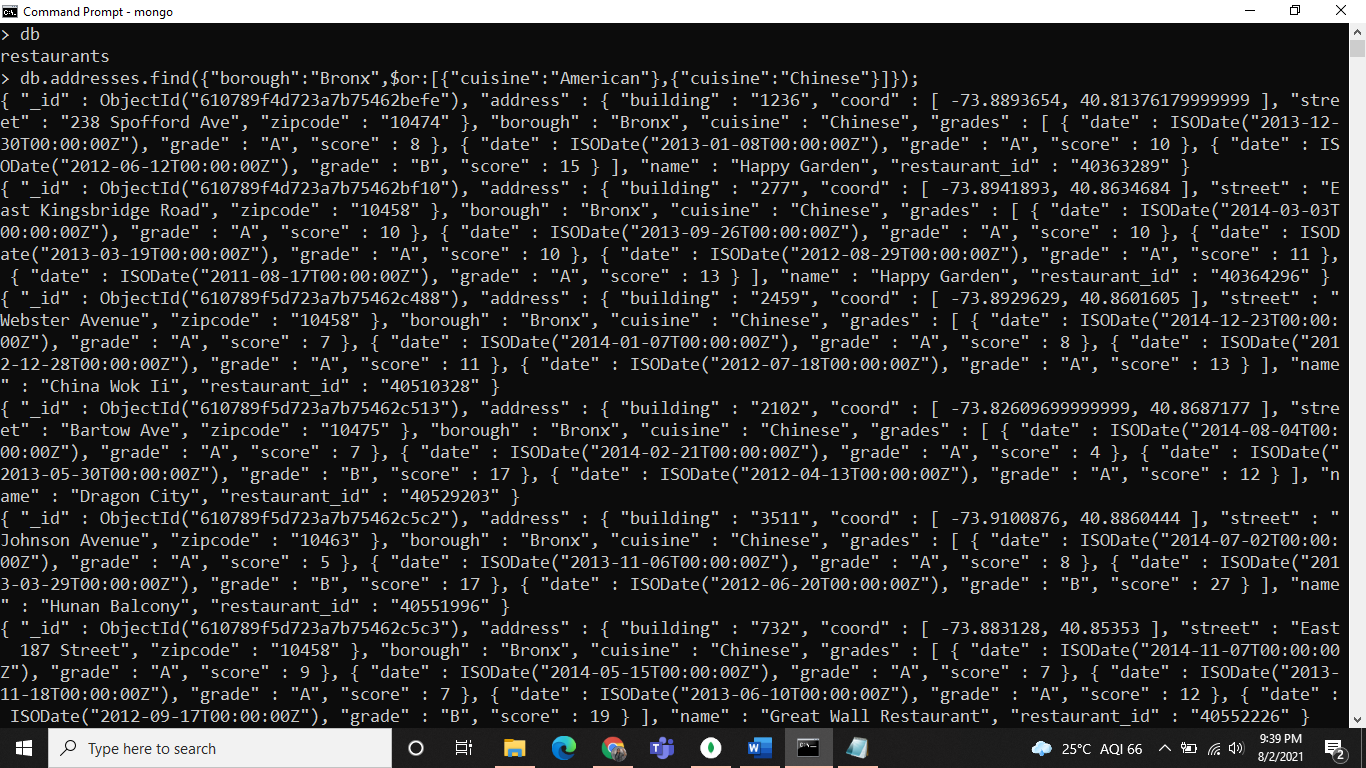
Q.15



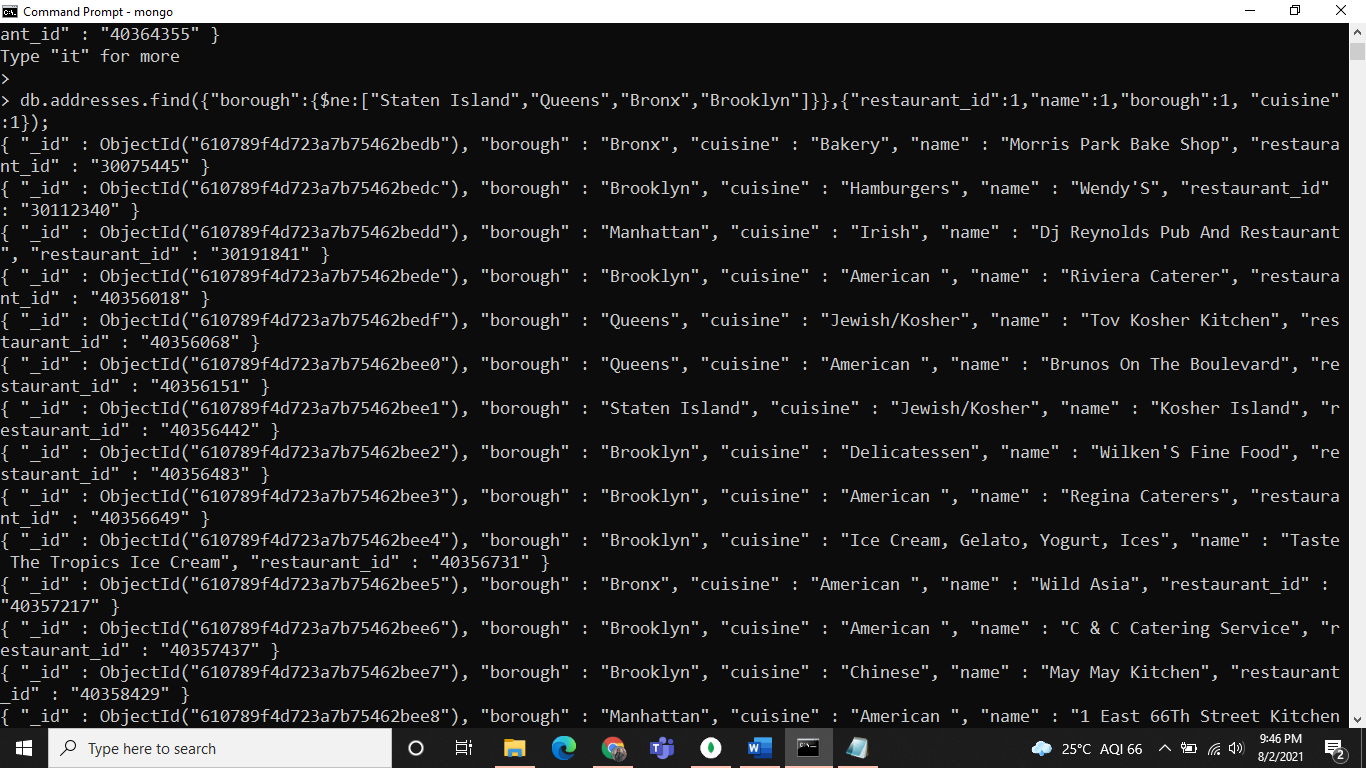
Q.16



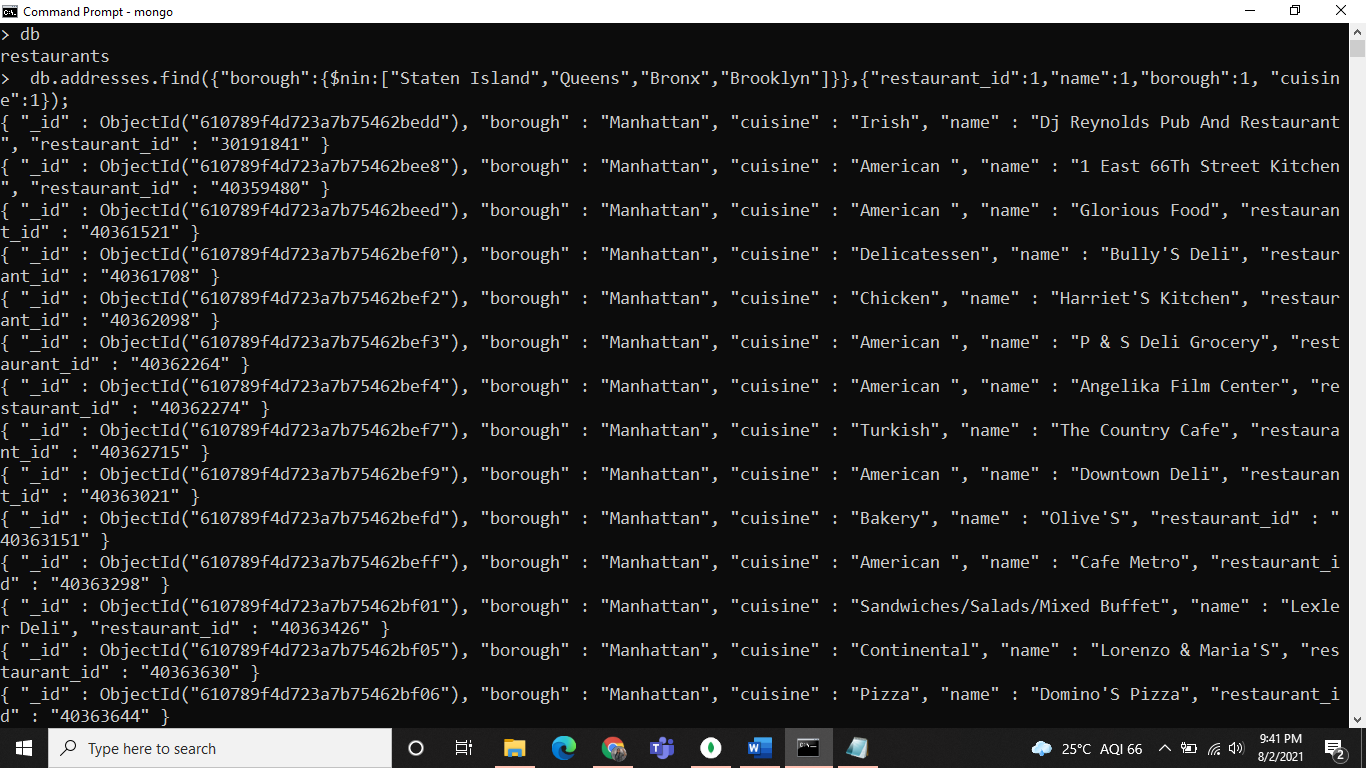
Q.17



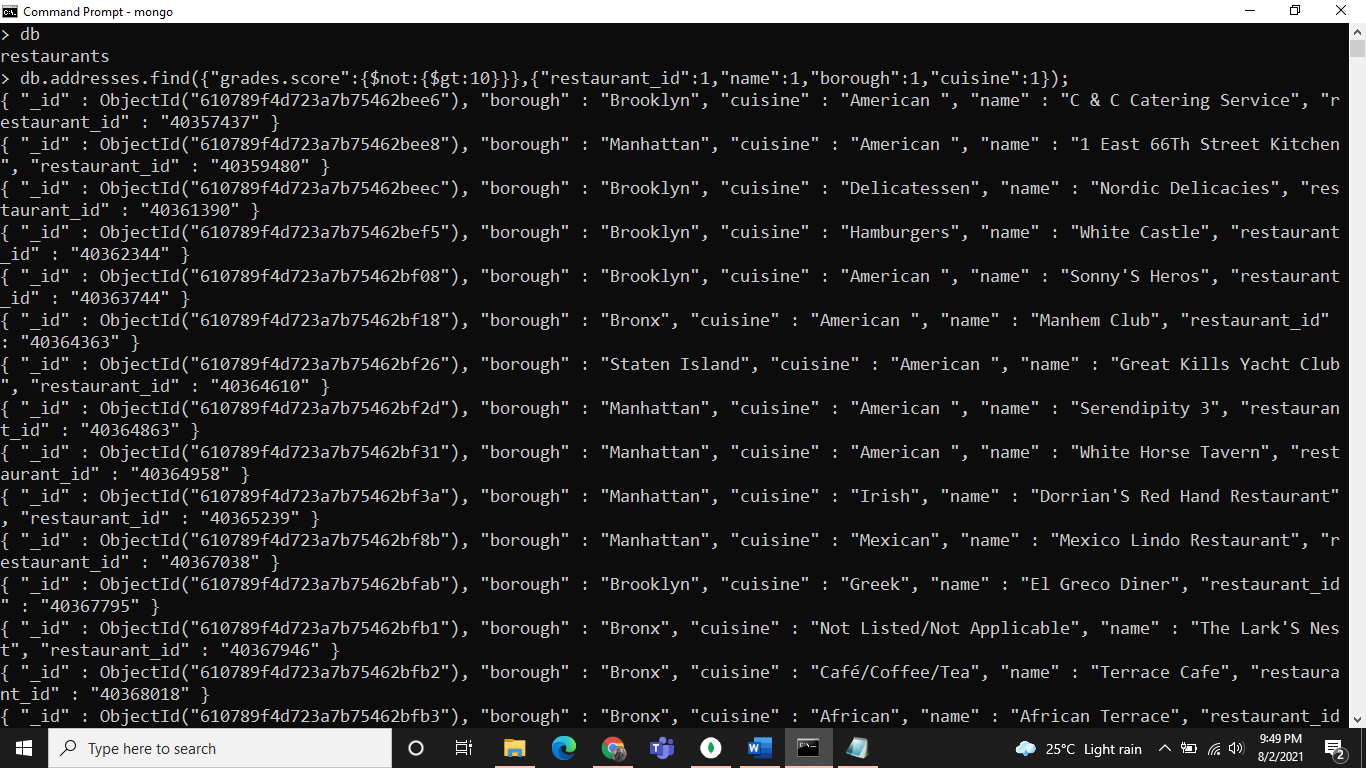
Q.18



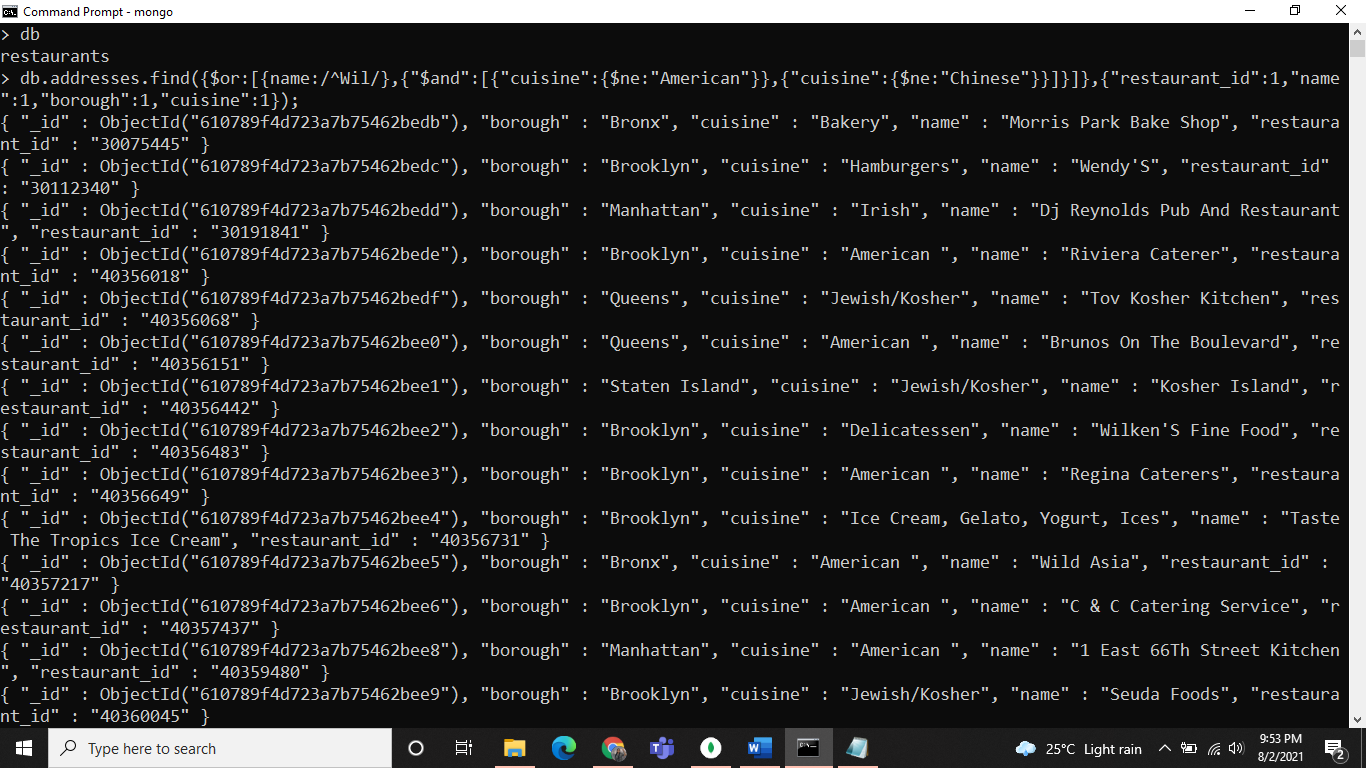
Q.19



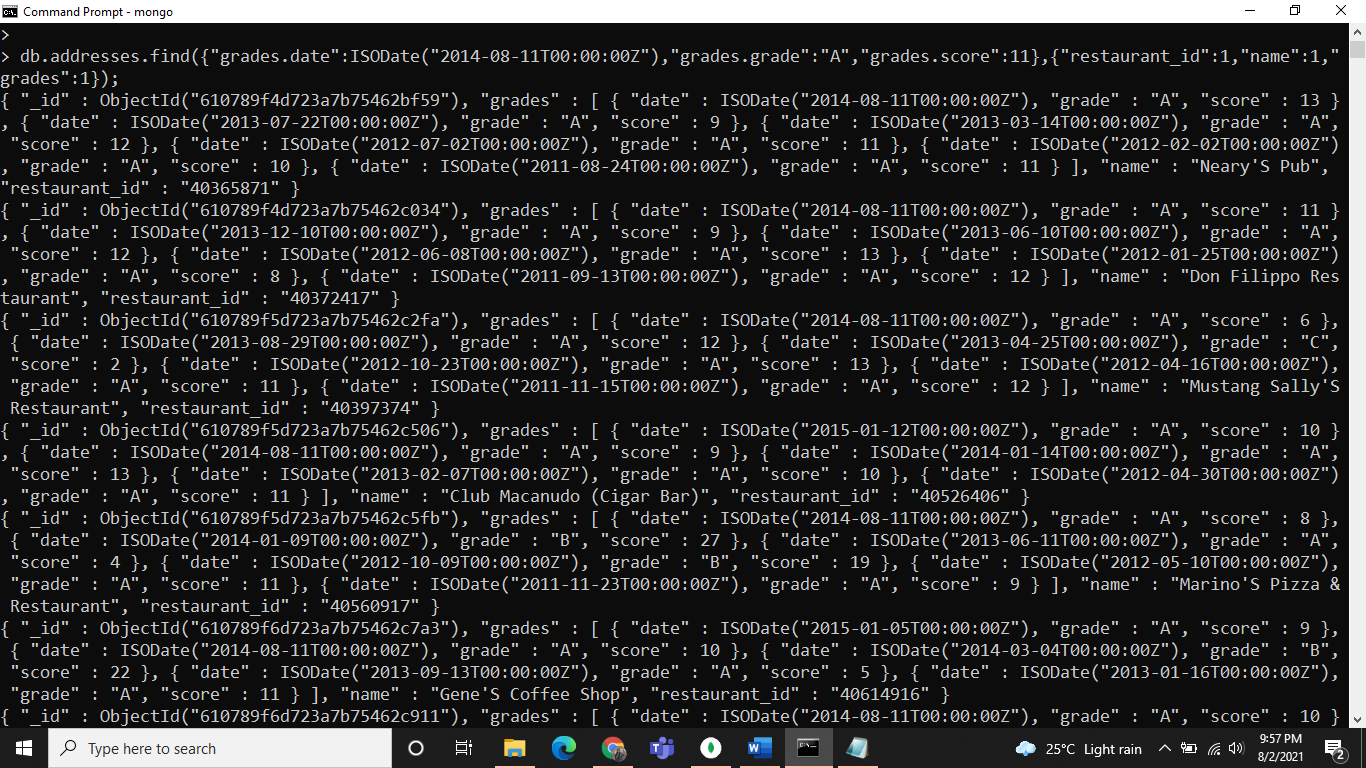
Q.20



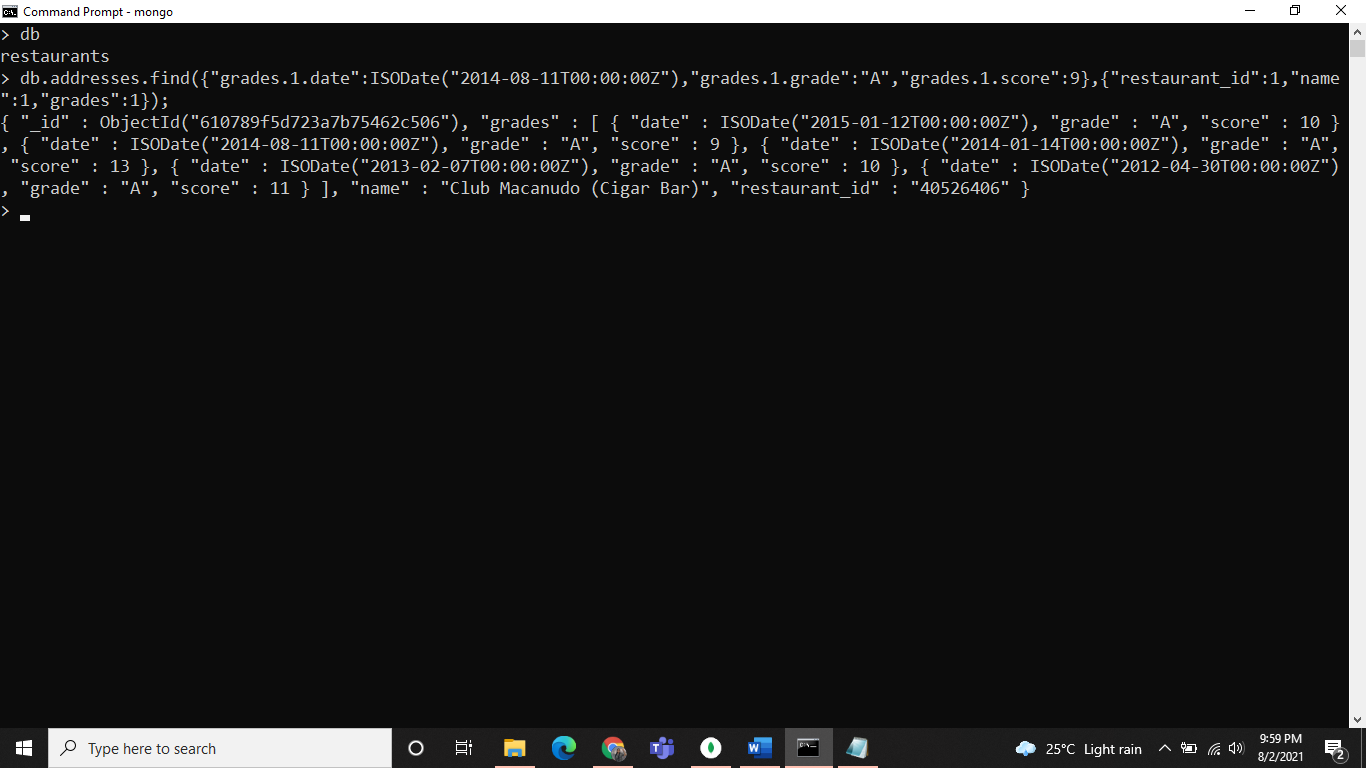
Q.21



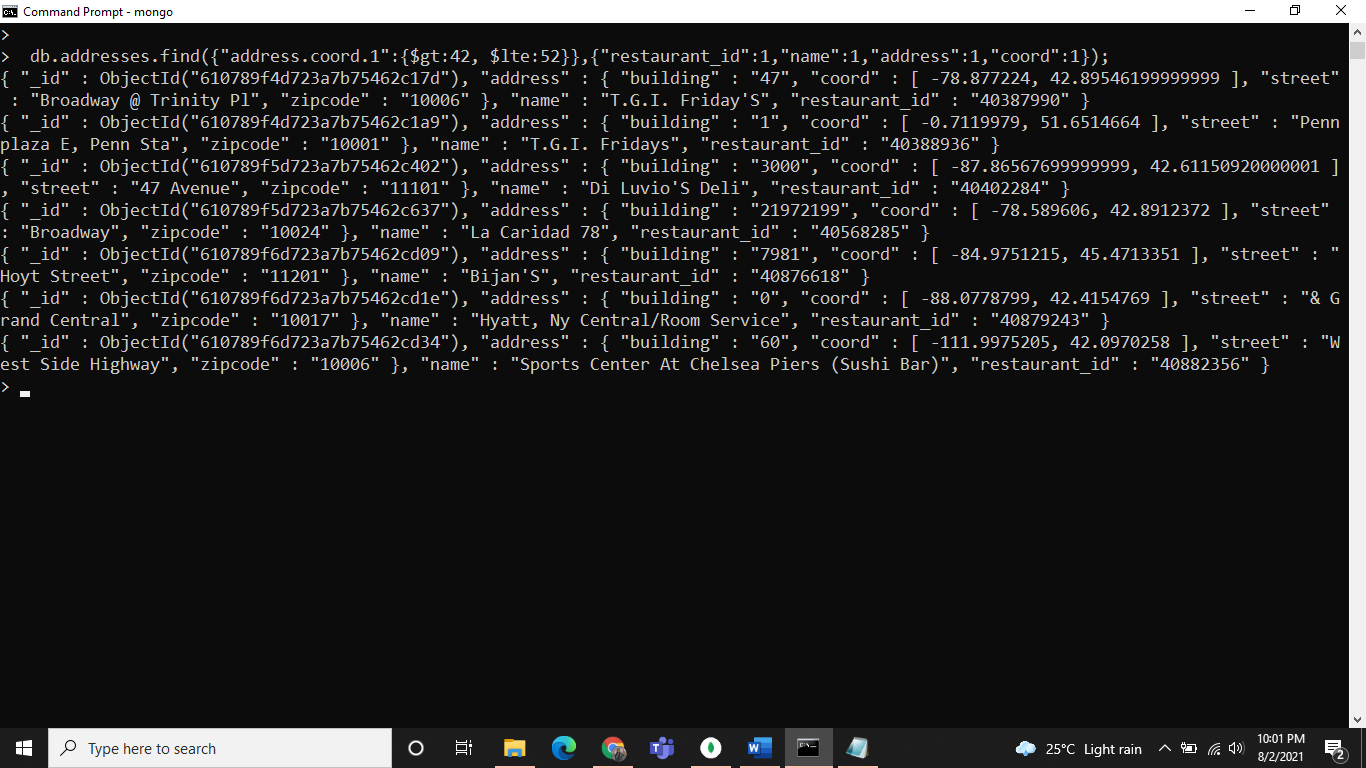
Q.22



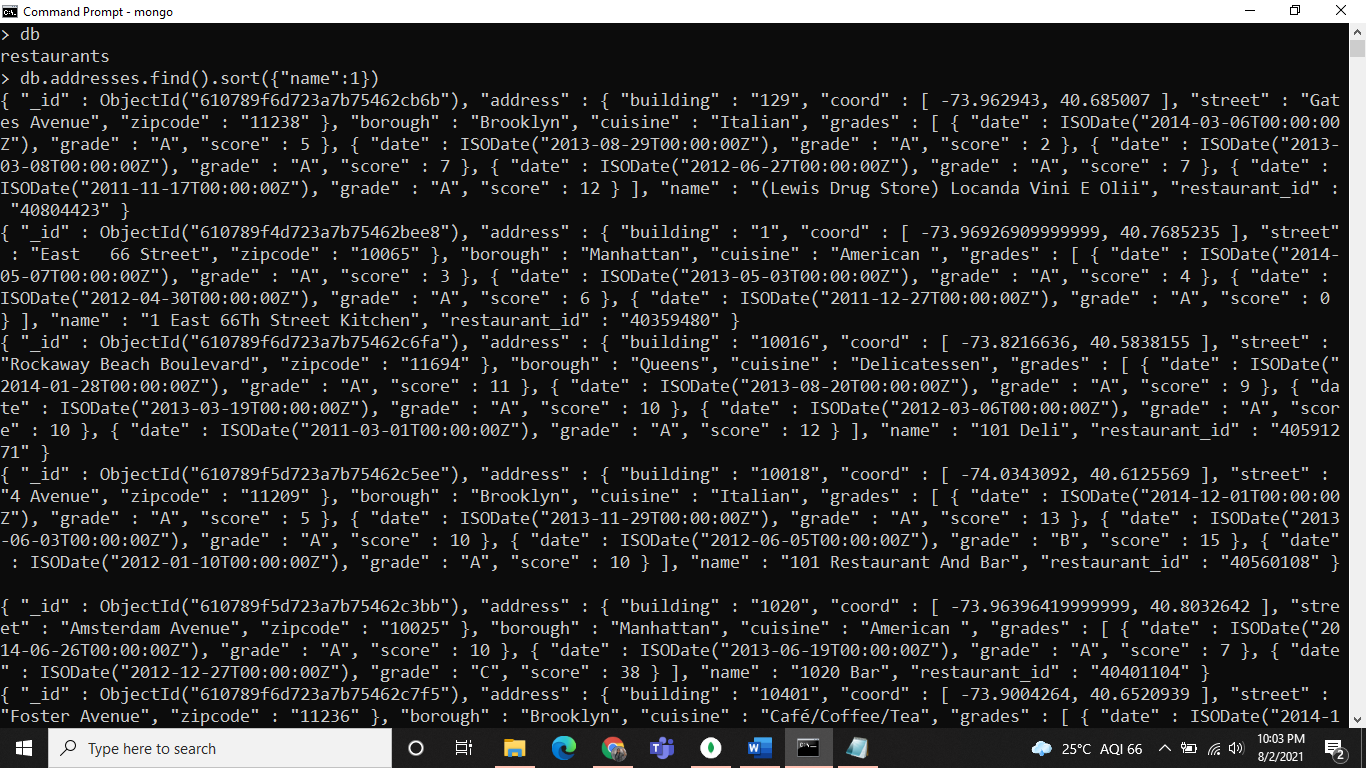
Q.23



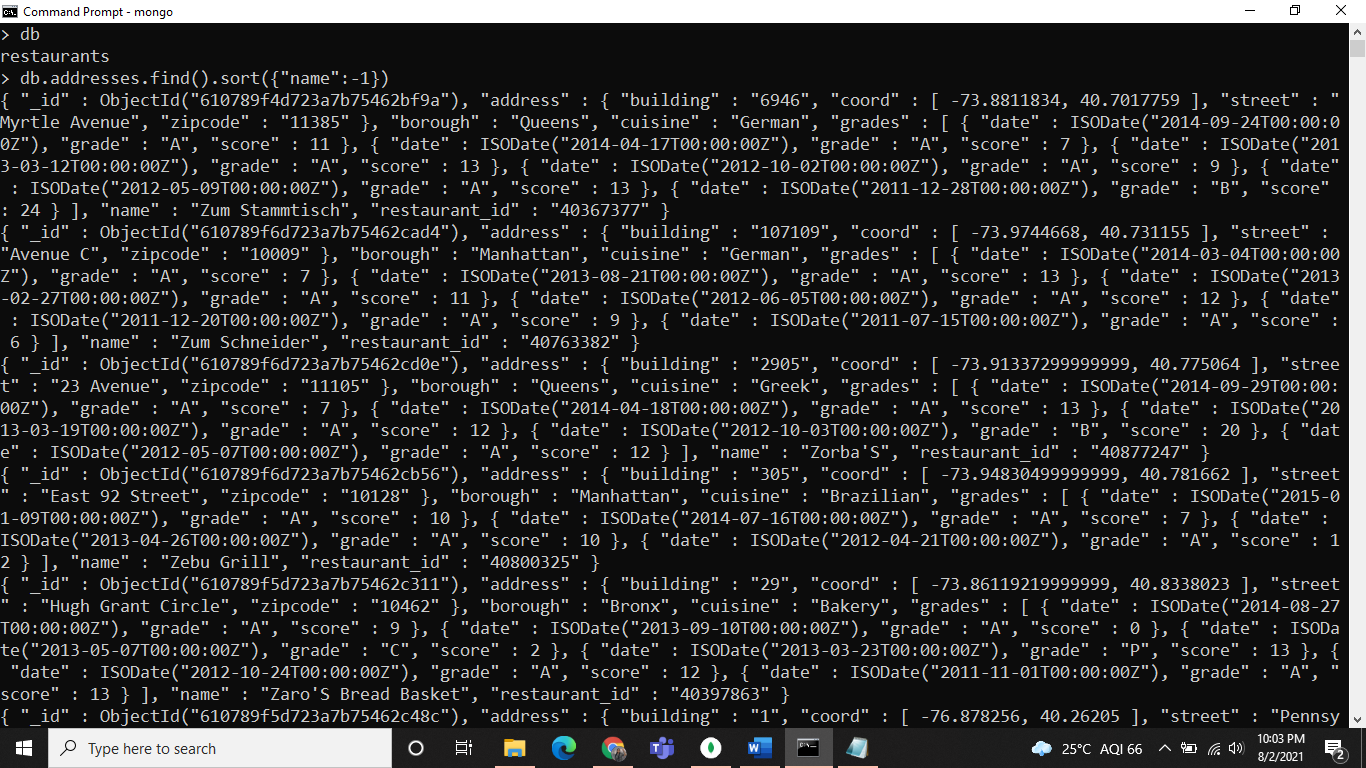
Q.24



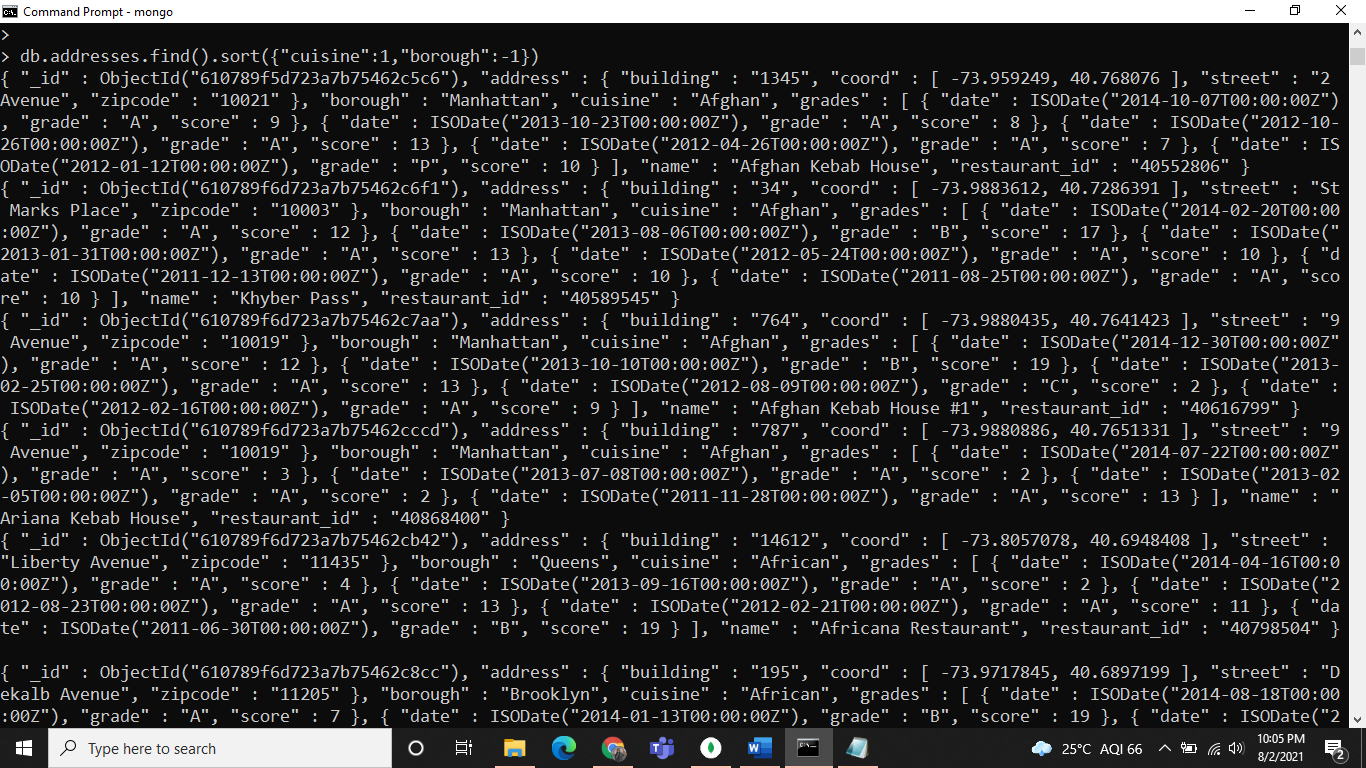
Q.25



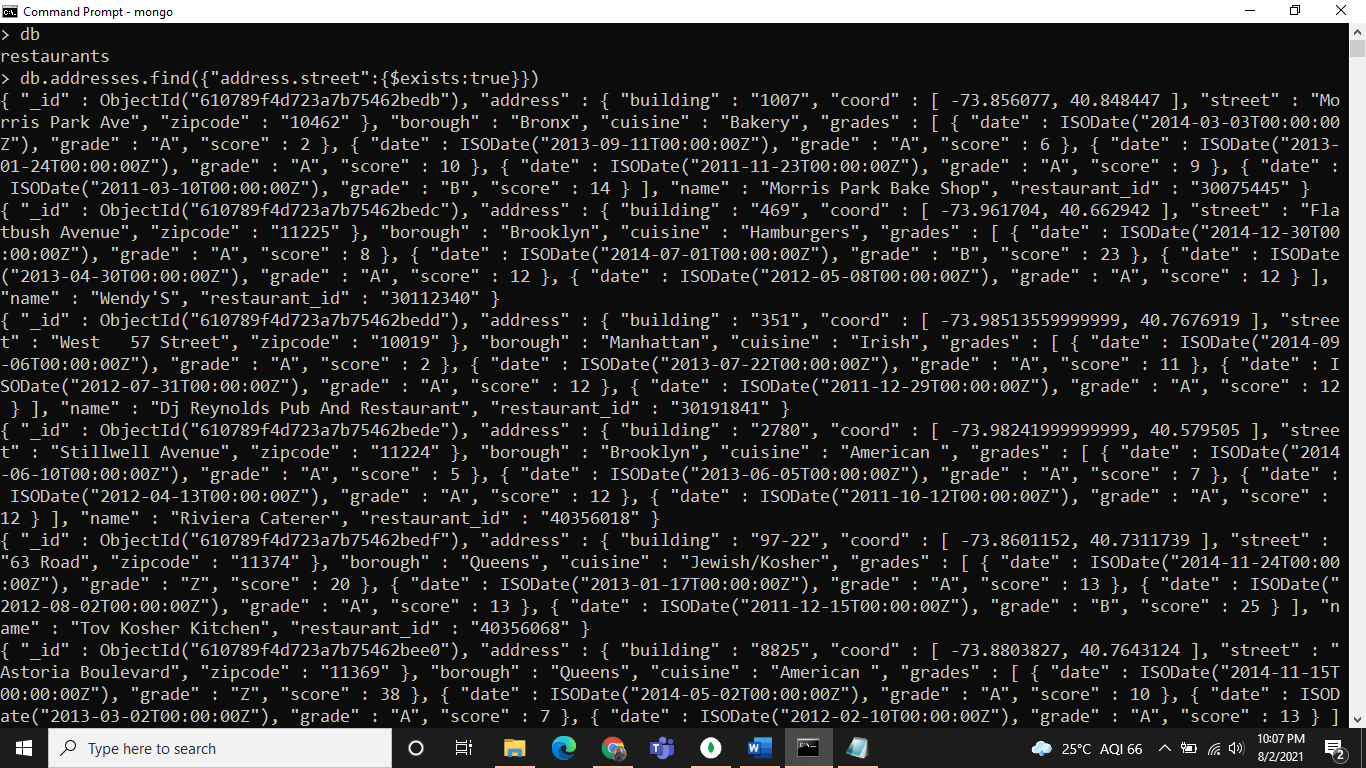
Q.26



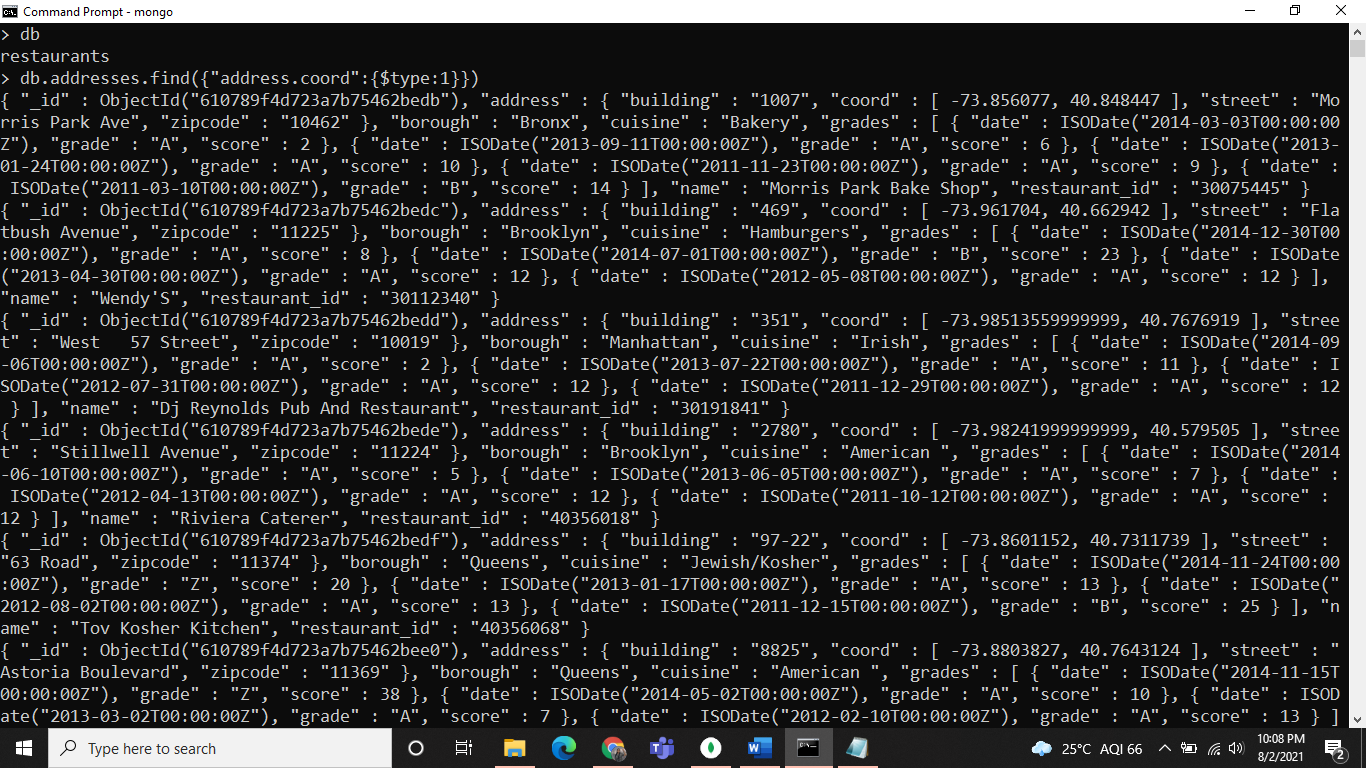
Q.27



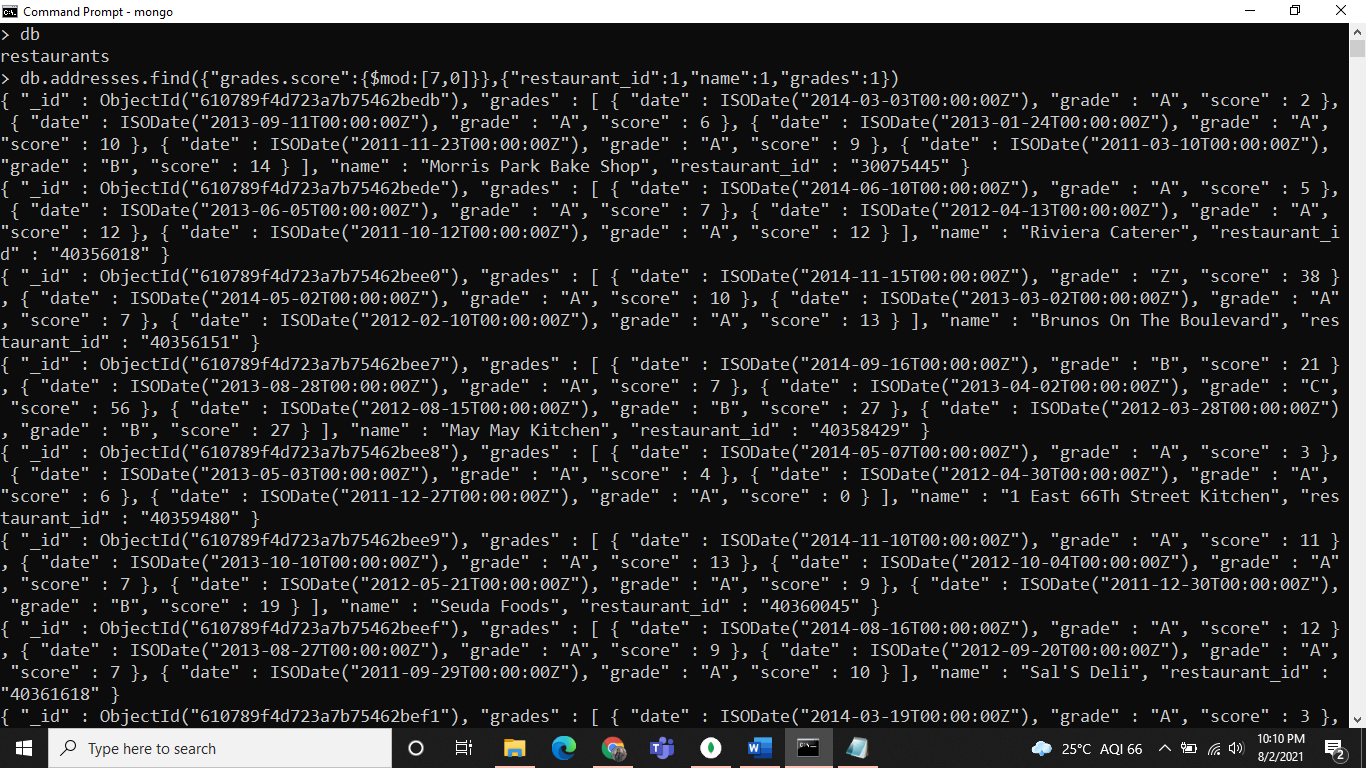
Q.28



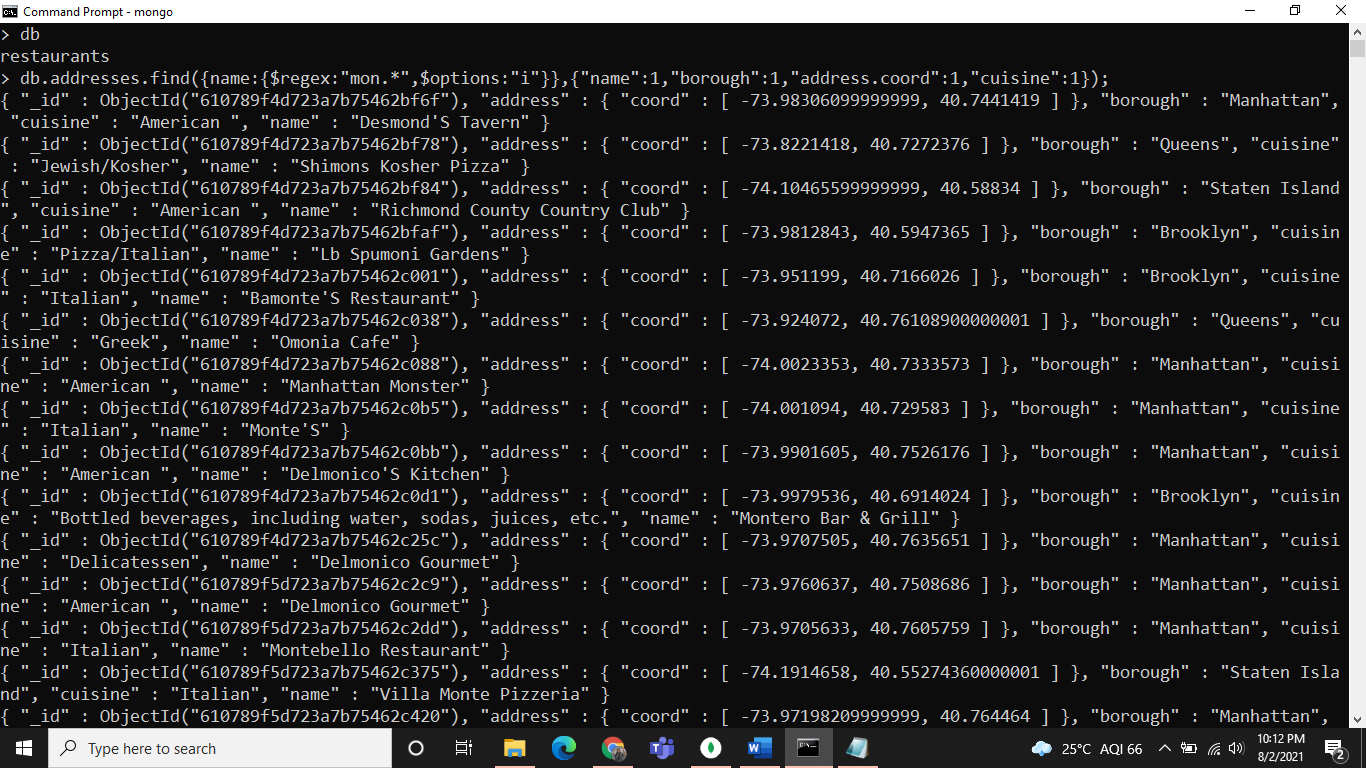
Q.29



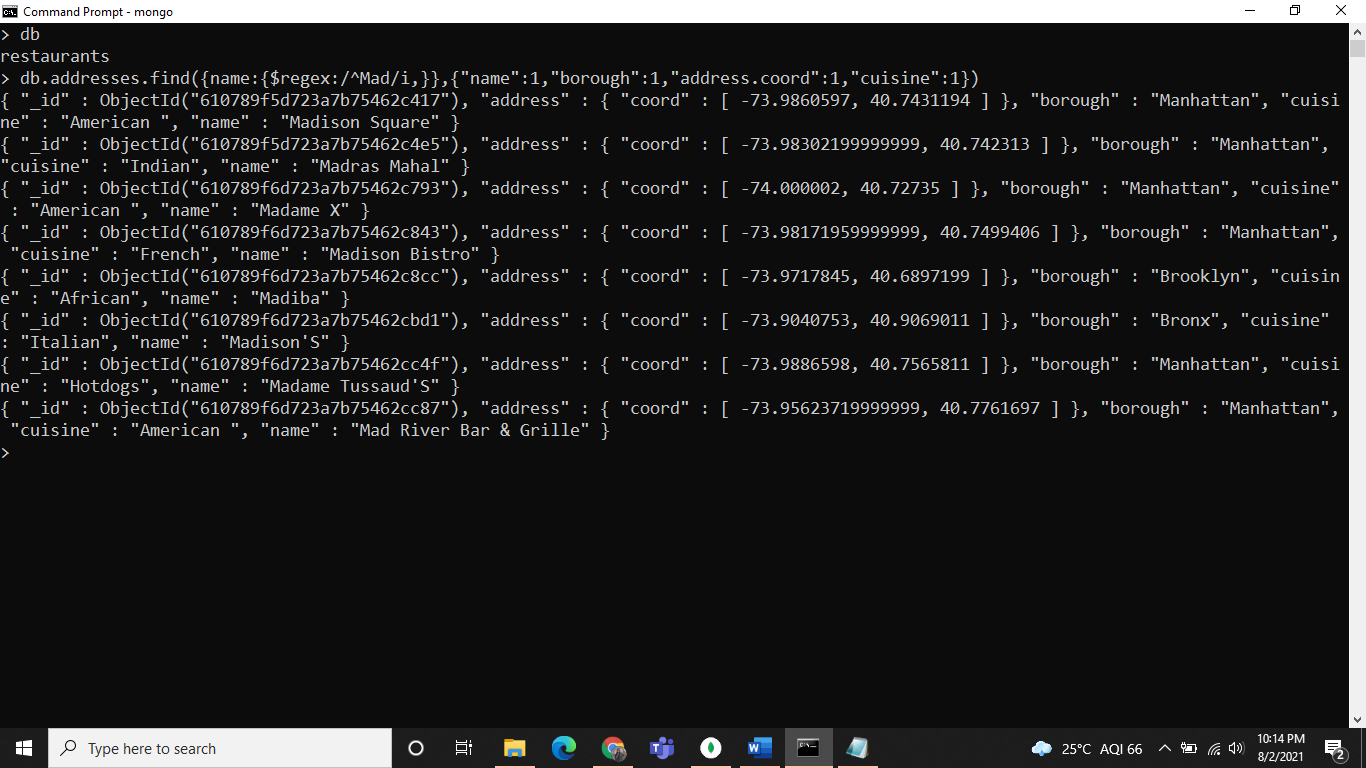
Q.30



Q.31



Q.32



-------------------------------------------------------------------------------------------------------------------------------