Spring 2017: NoSQL Team Blue Project Final Report

Table of Content

1. Team Name: BLUE

2. Team Members:

i. Rajat Kabra
ii. Cyrus Liang
iii. Isha Pradhan

3. Project Name

Crime Watch in Chicago

4. Project Description

Use Chicago's crime data set from 2001 to present and build a database application using MongoDB to garner meaningful insight into crime in Chicago.

5. Project Successes:

i. The data set we chose was taken from the data.gov website and hence was complete with no missing values. Battery, robbery, prostitution, etc all translate to real world issues that we face. Hence the analysis of this dataset proved fruitful as it helped us learn about the crime scene in various districts of Chicago.
ii. The use cases were designed in a way to uncover hidden information and relate crime aspects that were not self-evident.It was a success because we were all interested in getting a bigger understanding of the data set. This project allowed us to put a finger on the pulse of the city through NoSQL.
iii. We were able to finish the project in a timely manner due to proper communication between the group. We had multiple means of communication: in-person, phone messenger, Slack, Google group chat, WeChat, and email.
iv. Crime Watch in Chicago meets all the goals that Team BLUE set out to accomplish, so the product is a success.
v. There is valuable information that can be learned from the queries we designed.

6. Unexpected Events:

i. During the data importing there was a problem with the VM memory. By default, the VM is given memory of 512 MB. During importing the data into mongos, we faced issue where socket refused accepting data. To overcome this issue, the memory of VM was increased to upto 4GB, depending on our machines.

7. Lessons Learned:

i. Through the project we learned about config server and shard server replication.
ii. We learned about importing the data into a sharded cluster.
iii. Establishing connection using PyMongo.
iv. Using high level language to query mongodb database.
v. Wrangling and importing a large dataset.
vi. Various aggregation queries.
vii. Using mongodb functions like sort, pretty, limit, match, group, gte, lte, regex, match, distinct, and string matching.
viii. Connecting multiple VM using NAT and host only adapter.
ix. Difference between mongo and mongos.

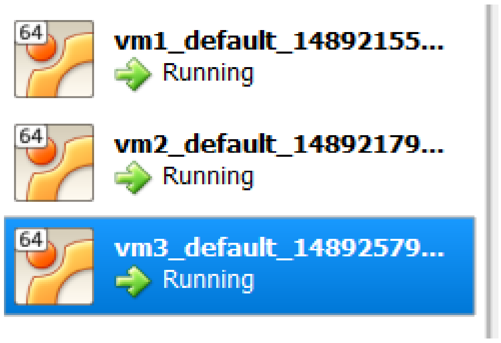
In future, we can try to work on a dataset which is a lot bigger than the dataset used. Instead of using local machine to connect to Python, we can try connecting the Python with nodes launched on AWS. We can also work on a bigger cluster and improve upon sharding strategy.

8. DataSet:

i. https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD
ii. This dataset contains 22 columns, has over 65,000 records/rows of data, is a total size of 1.38GB, and reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to present, minus the most recent seven days.
iii. Run these two command in Ubuntu:
   - wget -o crime.csv “https://data.cityofchicago.org/api/views/ijzp-q8t2/rows.csv?accessType=DOWNLOAD”
   - mongoimport -d finalproject -c projectdb --drop --type csv --file crime.csv --headerline --host [host number] --port [port number]

9. NoSQL Configuration

i. Creating three virtual machine.



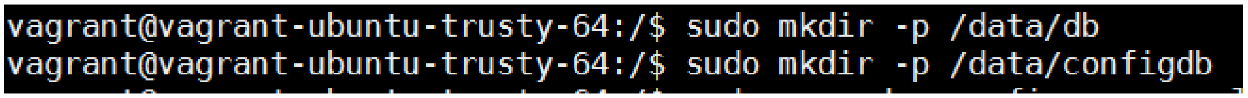
vm1_default_14892155...
Running

vm2_default_14892179...
Running

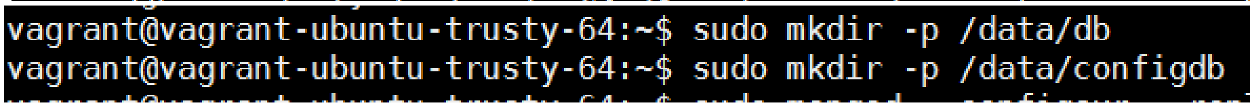
vm3_default_14892579...
Running

ii. MongoDB was installed on all the virtual machines.

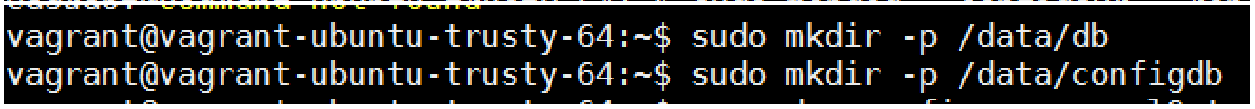
iii. A data folder was created on all three VM. In those data folder, two directories were created, one for config server (configdb) and one for shard server (db). The same process was done on all the VMs.



```
vagrant@vagrant-ubuntu-trusty-64:/$ sudo mkdir -p /data/db
vagrant@vagrant-ubuntu-trusty-64:/$ sudo mkdir -p /data/configdb
```

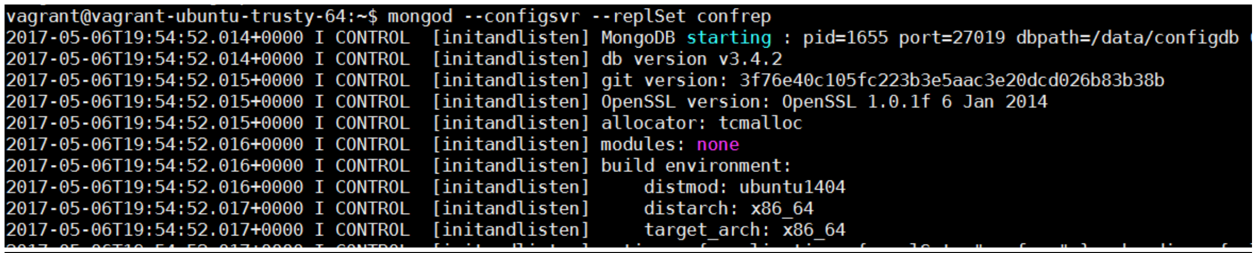


```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mkdir -p /data/db
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mkdir -p /data/configdb
```

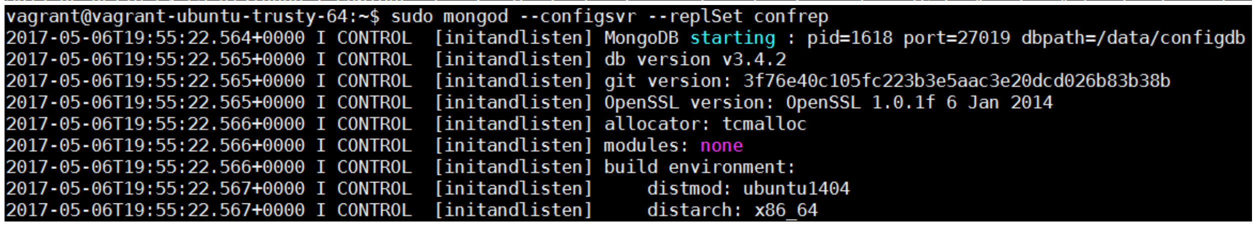


```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mkdir -p /data/db
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mkdir -p /data/configdb
```

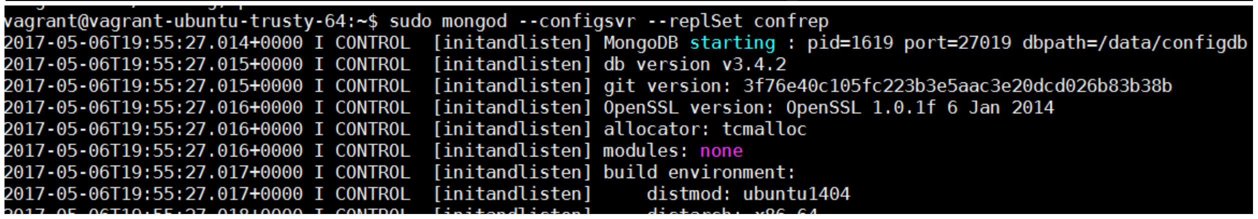
iv. Configuration server was started on all the three virtual machines. They were started as a part of ‘confrep’ replica set.



```
vagrant@vagrant-ubuntu-trusty-64:~$ mongod --configsvr --replSet confrep
2017-05-06T19:54:52.014+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1655 port=27019 dbpath=/data/configdb
2017-05-06T19:54:52.014+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T19:54:52.015+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T19:54:52.015+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T19:54:52.015+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-05-06T19:54:52.016+0000 I CONTROL  [initandlisten] modules: none
2017-05-06T19:54:52.016+0000 I CONTROL  [initandlisten] build environment:
2017-05-06T19:54:52.016+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-05-06T19:54:52.017+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-05-06T19:54:52.017+0000 I CONTROL  [initandlisten]     target_arch: x86_64
```

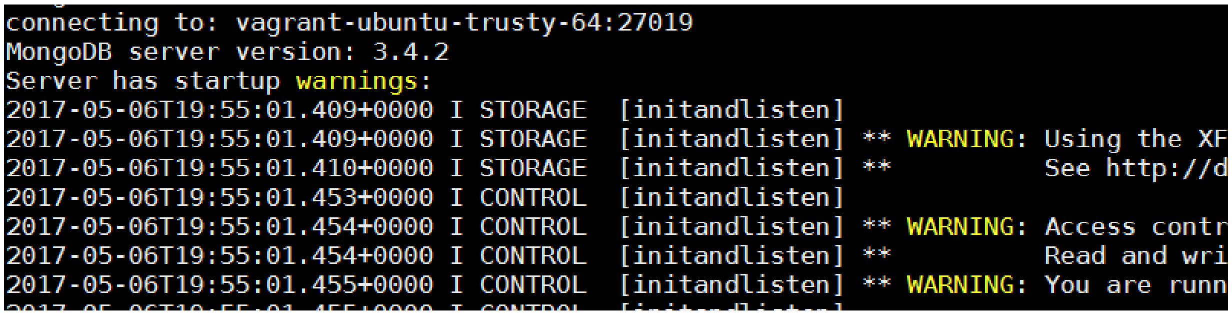


```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mongod --configsvr --replSet confrep
2017-05-06T19:55:22.564+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1618 port=27019 dbpath=/data/configdb
2017-05-06T19:55:22.565+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T19:55:22.565+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T19:55:22.565+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T19:55:22.566+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-05-06T19:55:22.566+0000 I CONTROL  [initandlisten] modules: none
2017-05-06T19:55:22.566+0000 I CONTROL  [initandlisten] build environment:
2017-05-06T19:55:22.567+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-05-06T19:55:22.567+0000 I CONTROL  [initandlisten]     distarch: x86_64
```



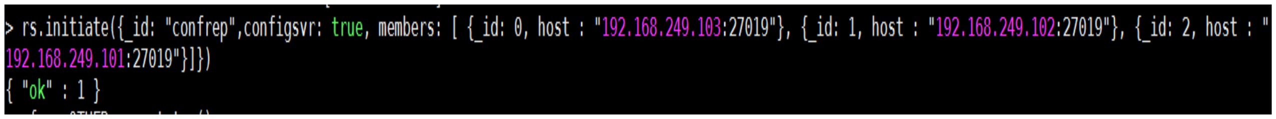
```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mongod --configsvr --replSet confrep
2017-05-06T19:55:27.014+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1619 port=27019 dbpath=/data/configdb
2017-05-06T19:55:27.015+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T19:55:27.015+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T19:55:27.016+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T19:55:27.016+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-05-06T19:55:27.016+0000 I CONTROL  [initandlisten] modules: none
2017-05-06T19:55:27.017+0000 I CONTROL  [initandlisten] build environment:
2017-05-06T19:55:27.017+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
```

v. We connect to mongo from host 192.168.249.103 and port 27019.



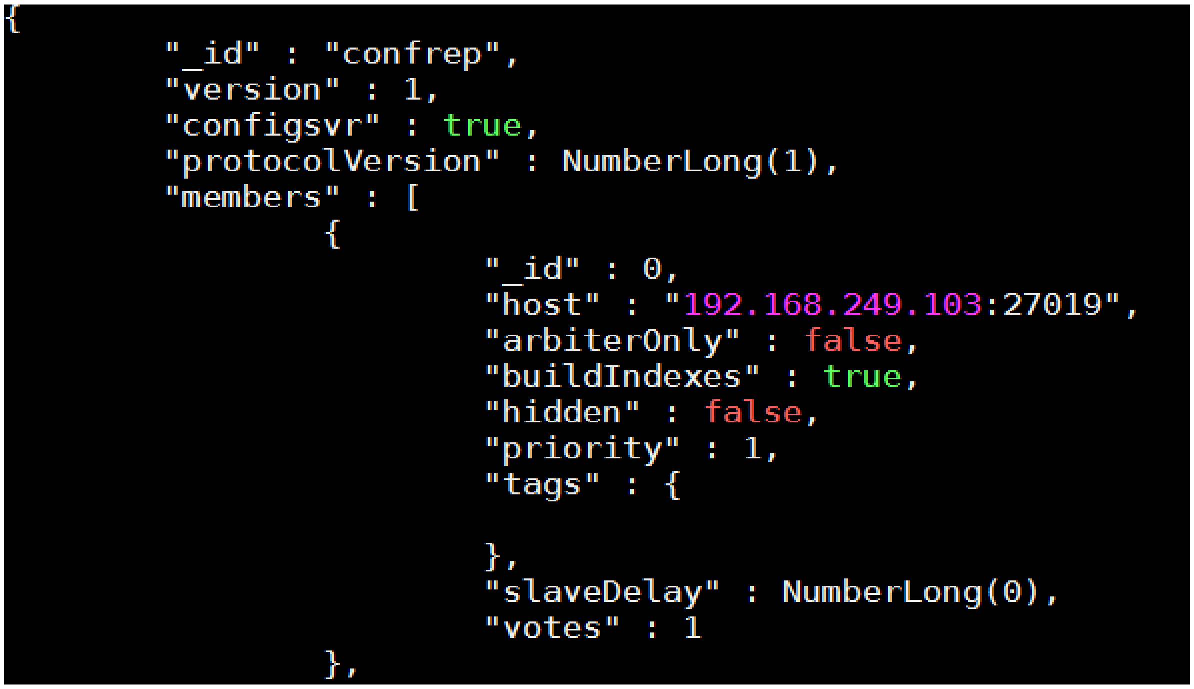
```
connecting to: vagrant-ubuntu-trusty-64:27019
MongoDB server version: 3.4.2
Server has startup warnings:
2017-05-06T19:55:01.409+0000 I STORAGE  [initandlisten]
2017-05-06T19:55:01.409+0000 I STORAGE  [initandlisten] ** WARNING: Using the XF
2017-05-06T19:55:01.410+0000 I STORAGE  [initandlisten] **          See http://d
2017-05-06T19:55:01.453+0000 I CONTROL  [initandlisten]
2017-05-06T19:55:01.454+0000 I CONTROL  [initandlisten] ** WARNING: Access contr
2017-05-06T19:55:01.454+0000 I CONTROL  [initandlisten] **          Read and wri
2017-05-06T19:55:01.455+0000 I CONTROL  [initandlisten] ** WARNING: You are runn
```

vi. All the three config servers are added.

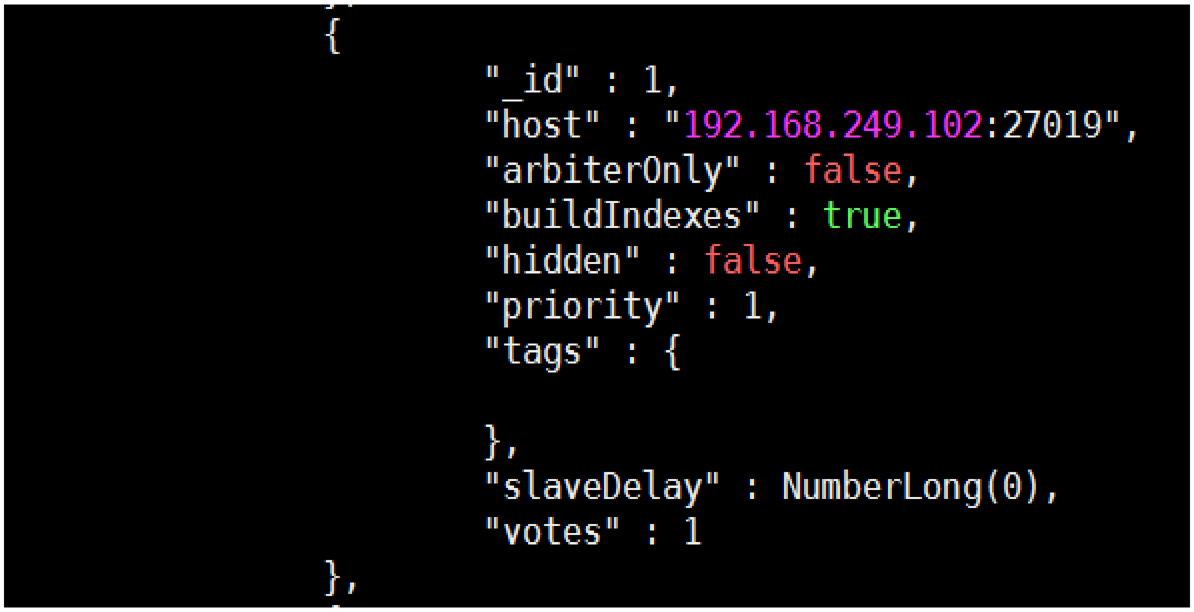


```
> rs.initiate({_id: "confrep",configsvr: true, members: [ {_id: 0, host : "192.168.249.103:27019"}, {_id: 1, host : "192.168.249.102:27019"}, {_id: 2, host : "
192.168.249.101:27019"}]})
{ "ok" : 1 }
```

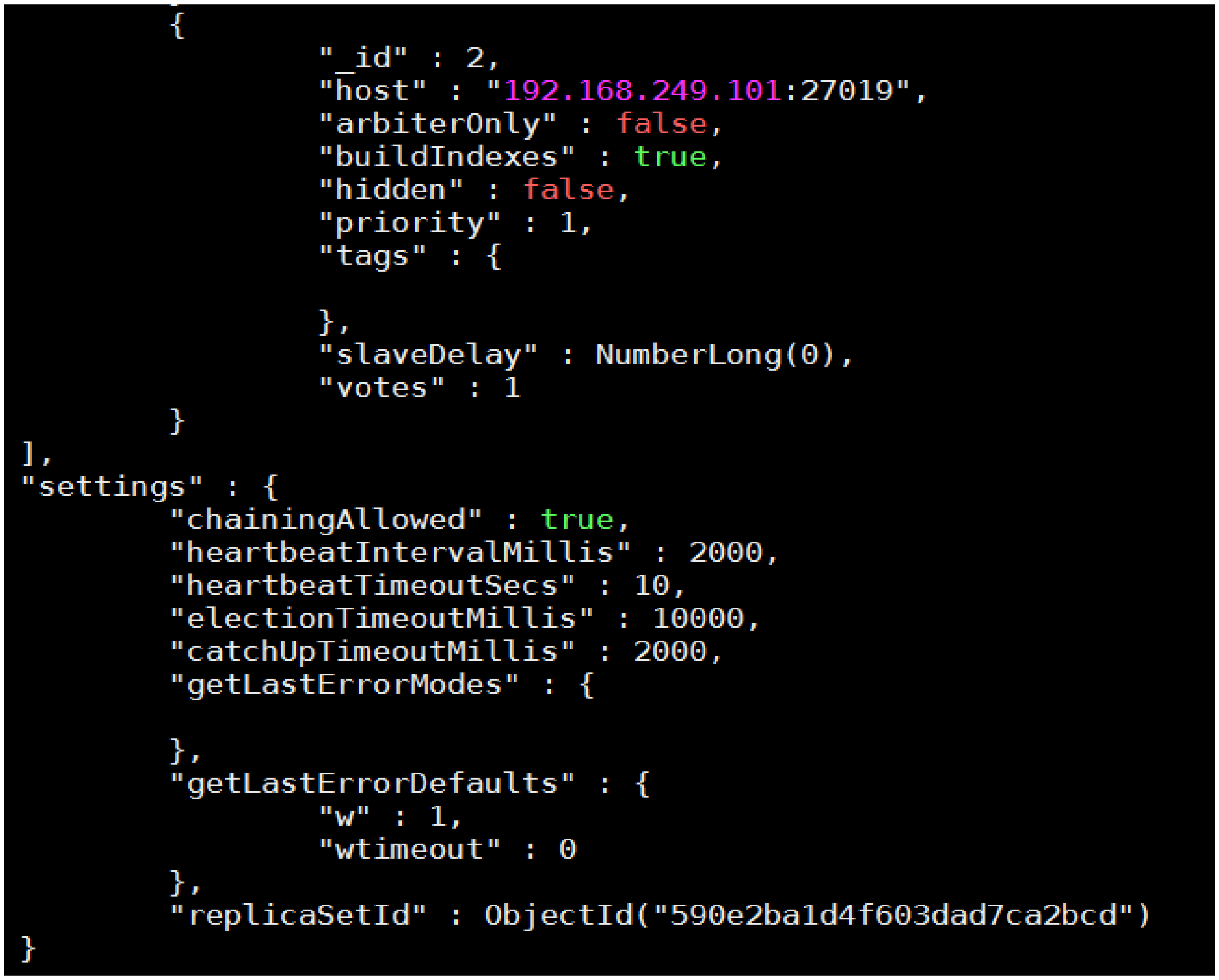
vii. Status after adding config servers.



```
{
        "_id" : "confrep",
        "version" : 1,
        "configsvr" : true,
        "protocolVersion" : NumberLong(1),
        "members" : [
                {
                        "_id" : 0,
                        "host" : "192.168.249.103:27019",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
```

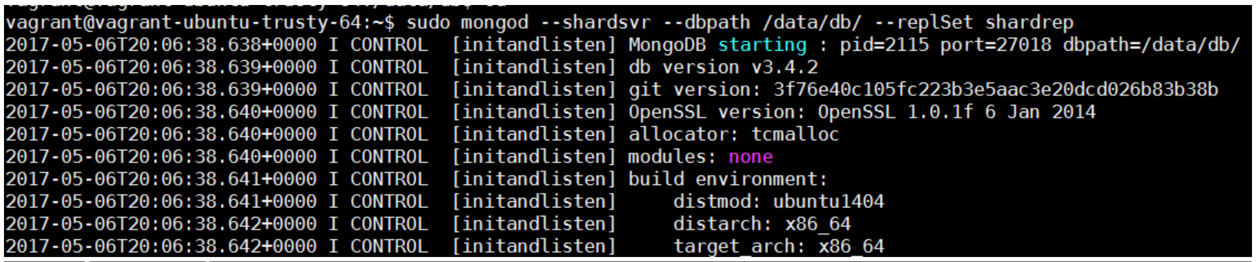


```
                {
                        "_id" : 1,
                        "host" : "192.168.249.102:27019",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
```

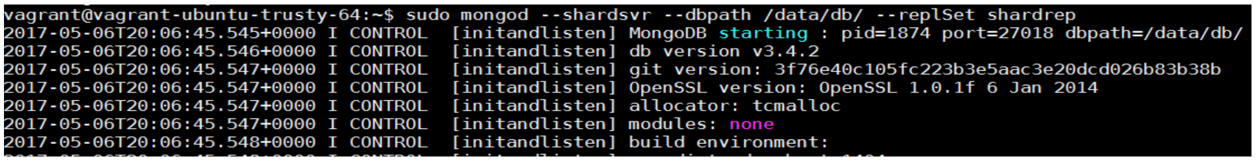


```
        {
                "_id" : 2,
                "host" : "192.168.249.101:27019",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        }
],
"settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : 2000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
                "w" : 1,
                "wtimeout" : 0
        },
        "replicaSetId" : ObjectId("590e2ba1d4f603dad7ca2bcd")
}
```

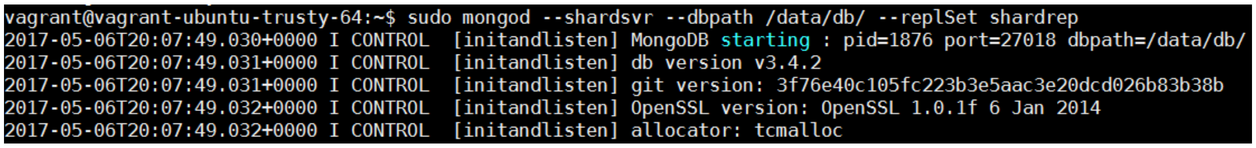
viii. The shard servers were started on every VM on port 27018 and as part of ‘shardrep’ replica set.



```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mongod --shardsvr --dbpath /data/db/ --replSet shardrep
2017-05-06T20:06:38.638+0000 I CONTROL  [initandlisten] MongoDB starting : pid=2115 port=27018 dbpath=/data/db/
2017-05-06T20:06:38.639+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T20:06:38.639+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T20:06:38.640+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T20:06:38.640+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-05-06T20:06:38.640+0000 I CONTROL  [initandlisten] modules: none
2017-05-06T20:06:38.641+0000 I CONTROL  [initandlisten] build environment:
2017-05-06T20:06:38.641+0000 I CONTROL  [initandlisten]     distmod: ubuntu1404
2017-05-06T20:06:38.642+0000 I CONTROL  [initandlisten]     distarch: x86_64
2017-05-06T20:06:38.642+0000 I CONTROL  [initandlisten]     target_arch: x86_64
```

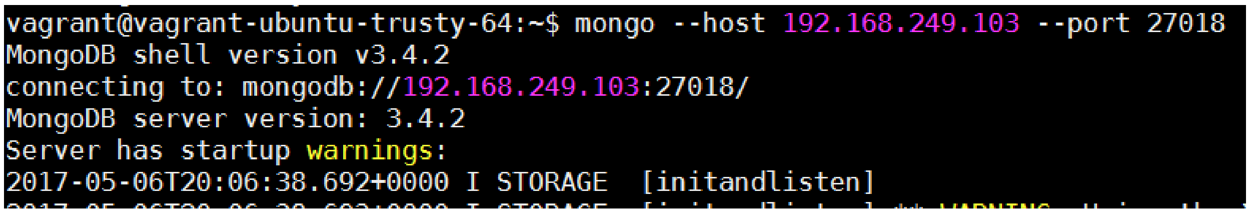


```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mongod --shardsvr --dbpath /data/db/ --replSet shardrep
2017-05-06T20:06:45.545+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1874 port=27018 dbpath=/data/db/
2017-05-06T20:06:45.546+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T20:06:45.547+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T20:06:45.547+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T20:06:45.547+0000 I CONTROL  [initandlisten] allocator: tcmalloc
2017-05-06T20:06:45.547+0000 I CONTROL  [initandlisten] modules: none
2017-05-06T20:06:45.548+0000 I CONTROL  [initandlisten] build environment:
```



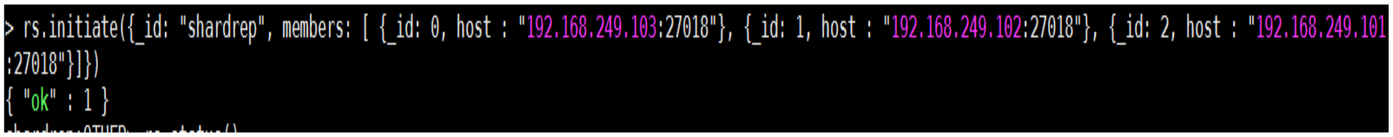
```
vagrant@vagrant-ubuntu-trusty-64:~$ sudo mongod --shardsvr --dbpath /data/db/ --replSet shardrep
2017-05-06T20:07:49.030+0000 I CONTROL  [initandlisten] MongoDB starting : pid=1876 port=27018 dbpath=/data/db/
2017-05-06T20:07:49.031+0000 I CONTROL  [initandlisten] db version v3.4.2
2017-05-06T20:07:49.031+0000 I CONTROL  [initandlisten] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T20:07:49.032+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T20:07:49.032+0000 I CONTROL  [initandlisten] allocator: tcmalloc
```

ix. Mongo was connected to the port 192.168.249.103 and port 27018 (Shard server).



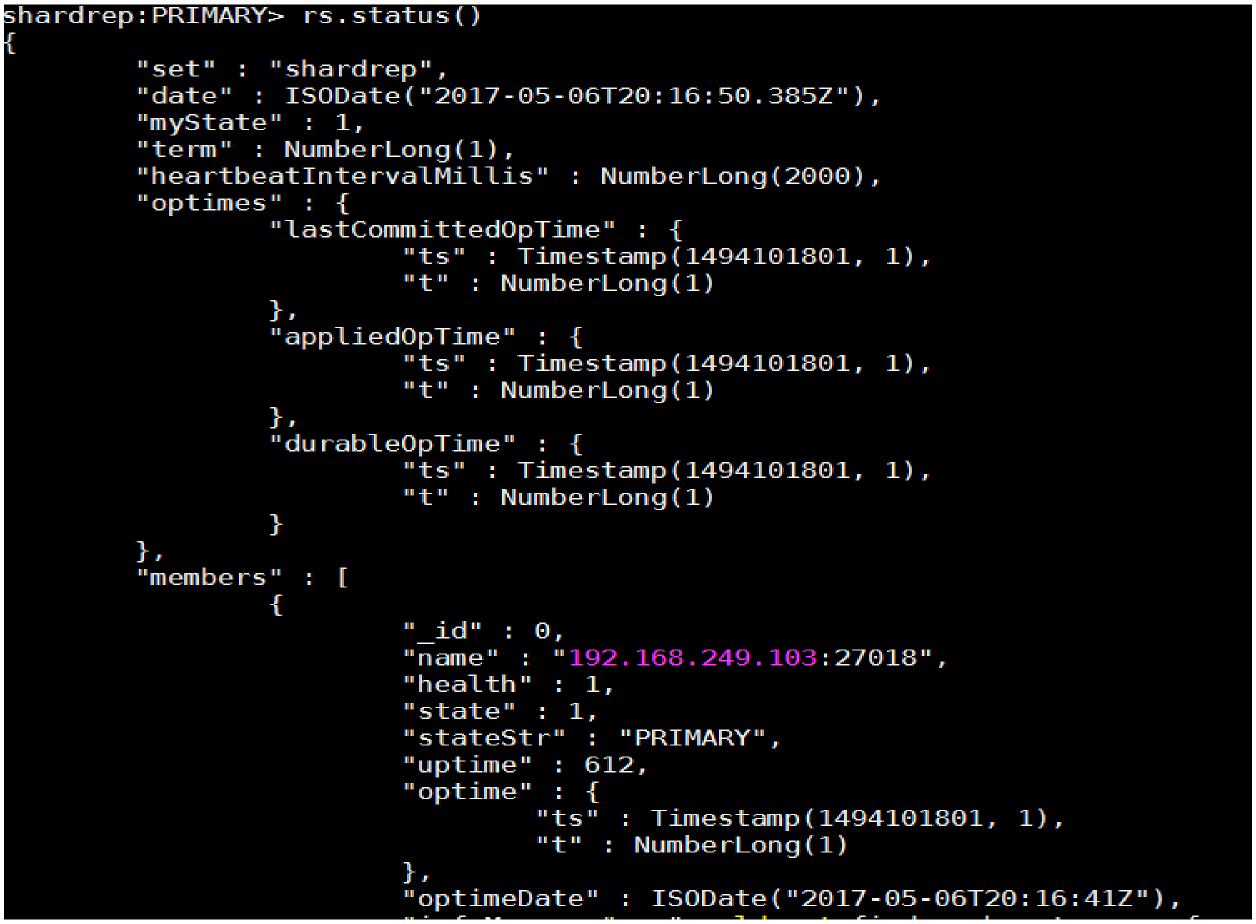
```
vagrant@vagrant-ubuntu-trusty-64:~$ mongo --host 192.168.249.103 --port 27018
MongoDB shell version v3.4.2
connecting to: mongodb://192.168.249.103:27018/
MongoDB server version: 3.4.2
Server has startup warnings:
2017-05-06T20:06:38.692+0000 I STORAGE  [initandlisten]
```

x. Shards server replica set is initiated.



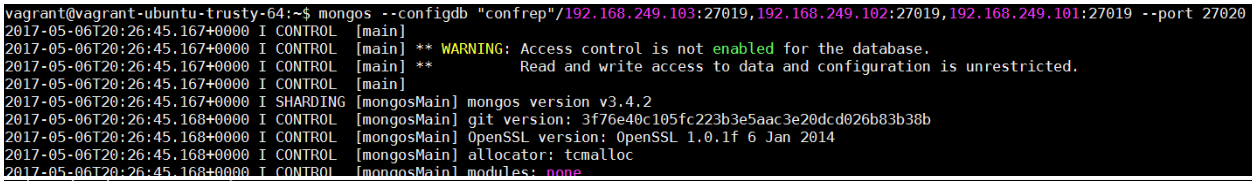
```
> rs.initiate({_id: "shardrep", members: [ {_id: 0, host : "192.168.249.103:27018"}, {_id: 1, host : "192.168.249.102:27018"}, {_id: 2, host : "192.168.249.101
:27018"}]})
{ "ok" : 1 }
```

xi. Status of shard server replica set.

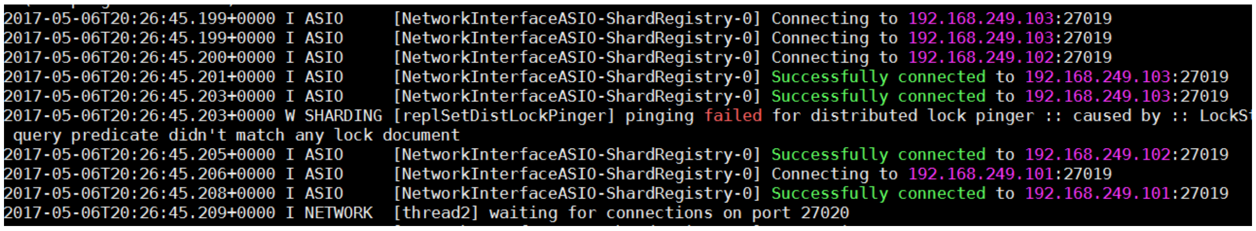


```
shardrep:PRIMARY> rs.status()
{
        "set" : "shardrep",
        "date" : ISODate("2017-05-06T20:16:50.385Z"),
        "myState" : 1,
        "term" : NumberLong(1),
        "heartbeatIntervalMillis" : NumberLong(2000),
        "optimes" : {
                "lastCommittedOpTime" : {
                        "ts" : Timestamp(1494101801, 1),
                        "t" : NumberLong(1)
                },
                "appliedOpTime" : {
                        "ts" : Timestamp(1494101801, 1),
                        "t" : NumberLong(1)
                },
                "durableOpTime" : {
                        "ts" : Timestamp(1494101801, 1),
                        "t" : NumberLong(1)
                }
        },
        "members" : [
                {
                        "_id" : 0,
                        "name" : "192.168.249.103:27018",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 612,
                        "optime" : {
                                "ts" : Timestamp(1494101801, 1),
                                "t" : NumberLong(1)
                        },
                        "optimeDate" : ISODate("2017-05-06T20:16:41Z"),
```

xii. Mongos is started on port 27020 and the config servers were added to it as a part of replica set ‘confrep’.

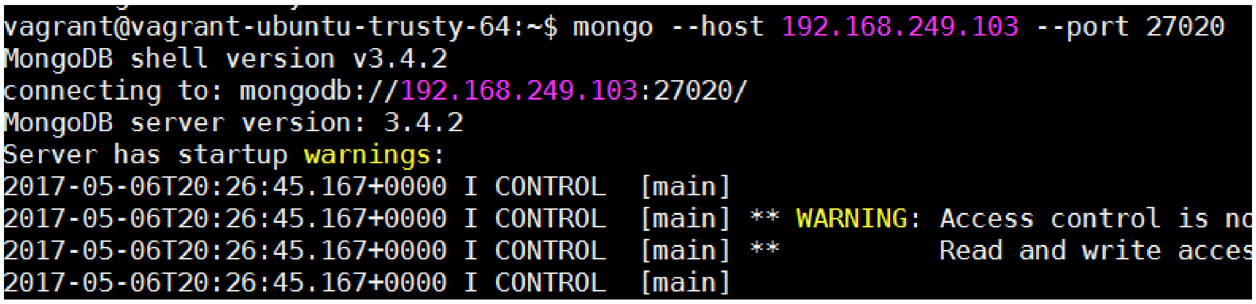


```
vagrant@vagrant-ubuntu-trusty-64:~$ mongos --configdb "confrep"/192.168.249.103:27019,192.168.249.102:27019,192.168.249.101:27019 --port 27020
2017-05-06T20:26:45.167+0000 I CONTROL  [main]
2017-05-06T20:26:45.167+0000 I CONTROL  [main] ** WARNING: Access control is not enabled for the database.
2017-05-06T20:26:45.167+0000 I CONTROL  [main] **          Read and write access to data and configuration is unrestricted.
2017-05-06T20:26:45.167+0000 I CONTROL  [main]
2017-05-06T20:26:45.167+0000 I SHARDING [mongosMain] mongos version v3.4.2
2017-05-06T20:26:45.168+0000 I CONTROL  [mongosMain] git version: 3f76e40c105fc223b3e5aac3e20dcd026b83b38b
2017-05-06T20:26:45.168+0000 I CONTROL  [mongosMain] OpenSSL version: OpenSSL 1.0.1f 6 Jan 2014
2017-05-06T20:26:45.168+0000 I CONTROL  [mongosMain] allocator: tcmalloc
2017-05-06T20:26:45.168+0000 I CONTROL  [mongosMain] modules: none
```

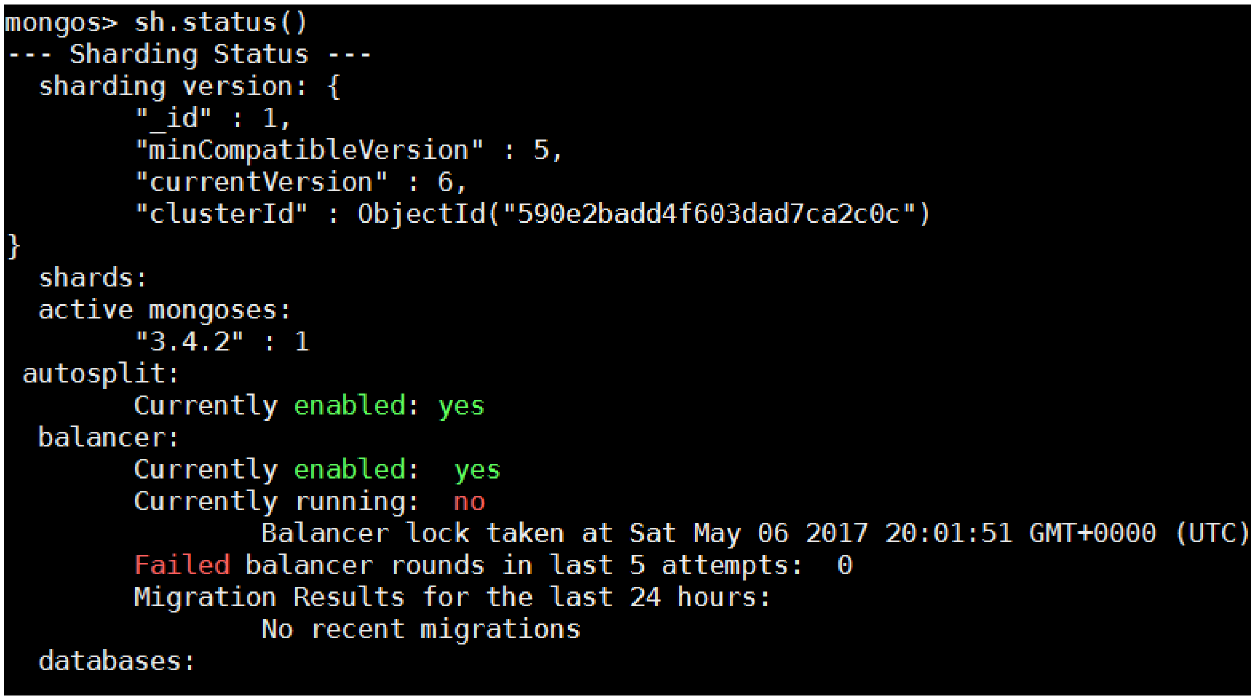


```
2017-05-06T20:26:45.199+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to 192.168.249.103:27019
2017-05-06T20:26:45.199+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to 192.168.249.103:27019
2017-05-06T20:26:45.200+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to 192.168.249.102:27019
2017-05-06T20:26:45.201+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully connected to 192.168.249.103:27019
2017-05-06T20:26:45.203+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully connected to 192.168.249.103:27019
2017-05-06T20:26:45.203+0000 W SHARDING [replSetDistLockPinger] pinging failed for distributed lock pinger :: caused by :: LockS
 query predicate didn't match any lock document
2017-05-06T20:26:45.205+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully connected to 192.168.249.102:27019
2017-05-06T20:26:45.206+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Connecting to 192.168.249.101:27019
2017-05-06T20:26:45.208+0000 I ASIO     [NetworkInterfaceASIO-ShardRegistry-0] Successfully connected to 192.168.249.101:27019
2017-05-06T20:26:45.209+0000 I NETWORK  [thread2] waiting for connections on port 27020
```

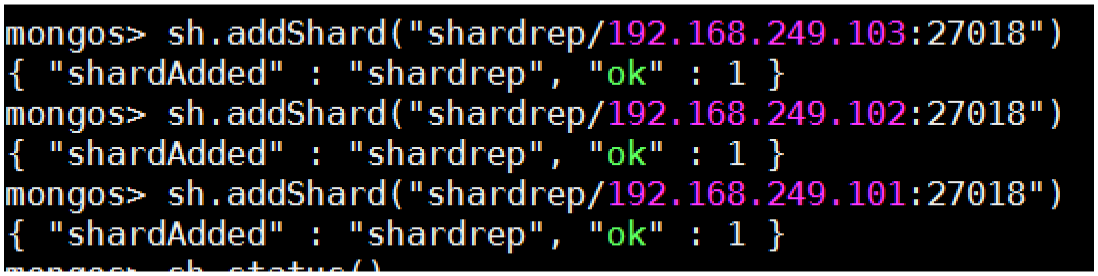
xiii. Connecting Mongo to Mongos.


```
vagrant@vagrant-ubuntu-trusty-64:~$ mongo --host 192.168.249.103 --port 27020
MongoDB shell version v3.4.2
connecting to: mongodb://192.168.249.103:27020/
MongoDB server version: 3.4.2
Server has startup warnings:
2017-05-06T20:26:45.167+0000 I CONTROL  [main]
2017-05-06T20:26:45.167+0000 I CONTROL  [main] ** WARNING: Access control is no
2017-05-06T20:26:45.167+0000 I CONTROL  [main] **          Read and write acces
2017-05-06T20:26:45.167+0000 I CONTROL  [main]
```

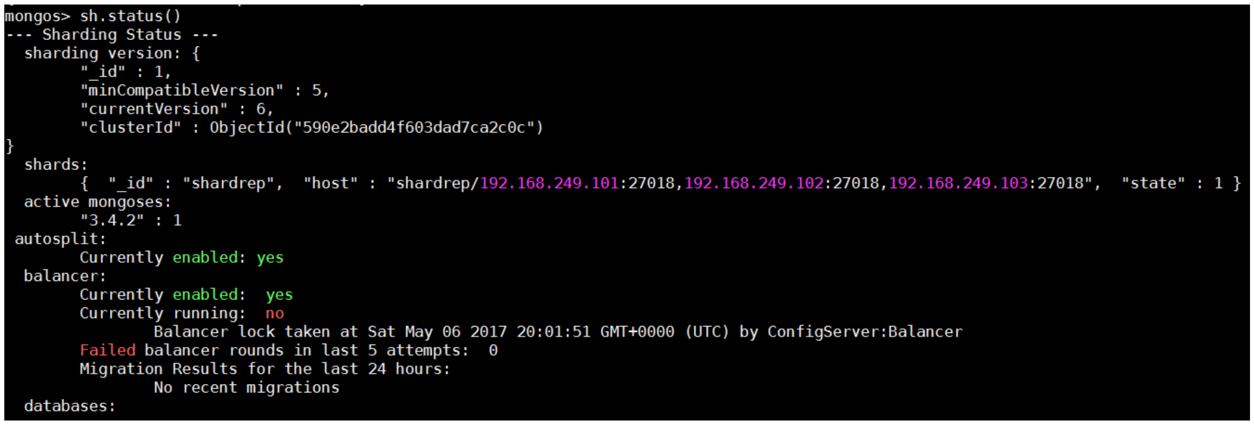
xiv. Status before adding shard servers.


```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("590e2badd4f603dad7ca2c0c")
}
  shards:
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Sat May 06 2017 20:01:51 GMT+0000 (UTC)
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
```

xv. Adding shards as part of ‘shardrep’ replica set.

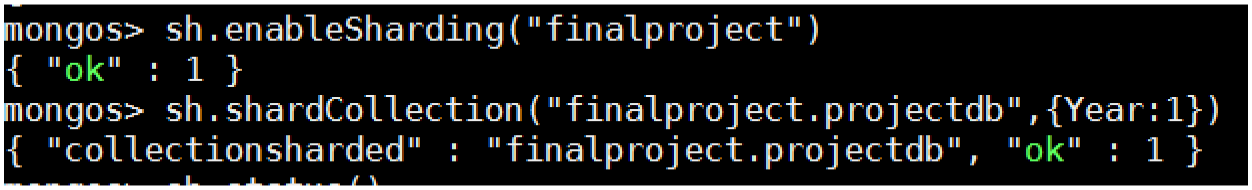

```
mongos> sh.addShard("shardrep/192.168.249.103:27018")
{ "shardAdded" : "shardrep", "ok" : 1 }
mongos> sh.addShard("shardrep/192.168.249.102:27018")
{ "shardAdded" : "shardrep", "ok" : 1 }
mongos> sh.addShard("shardrep/192.168.249.101:27018")
{ "shardAdded" : "shardrep", "ok" : 1 }
```

xvi. Status after adding shards but before enabling sharding.



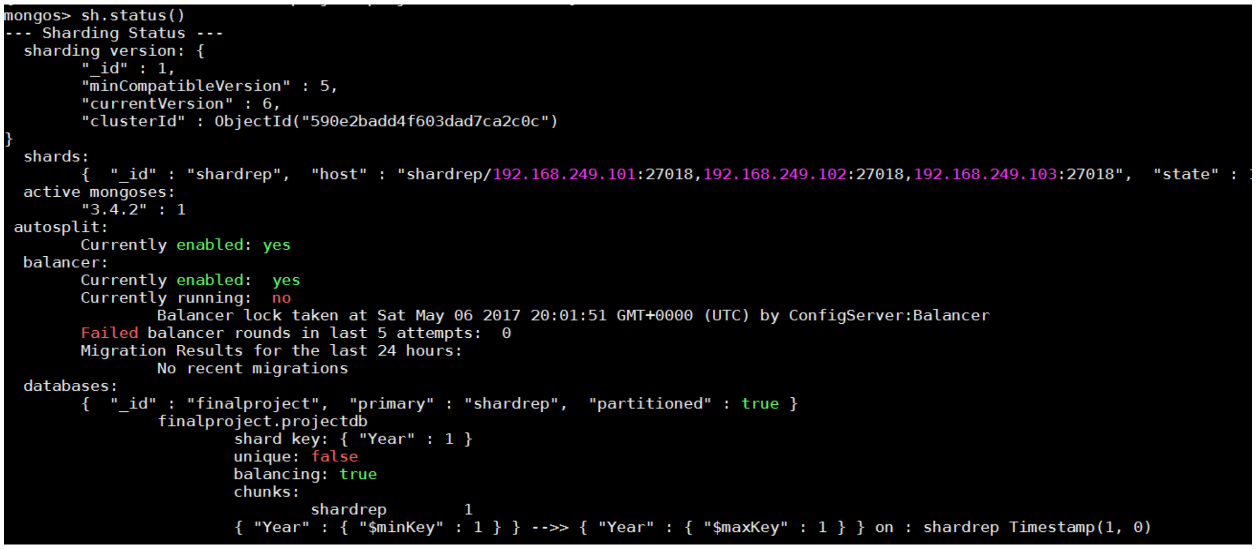
```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("590e2badd4f603dad7ca2c0c")
}
  shards:
        {  "_id" : "shardrep",  "host" : "shardrep/192.168.249.101:27018,192.168.249.102:27018,192.168.249.103:27018",  "state" : 1 }
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Sat May 06 2017 20:01:51 GMT+0000 (UTC) by ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
```

xvii. Enabling sharding on ‘finalproject’ database and project db collection using ‘Year’ as sharding key.



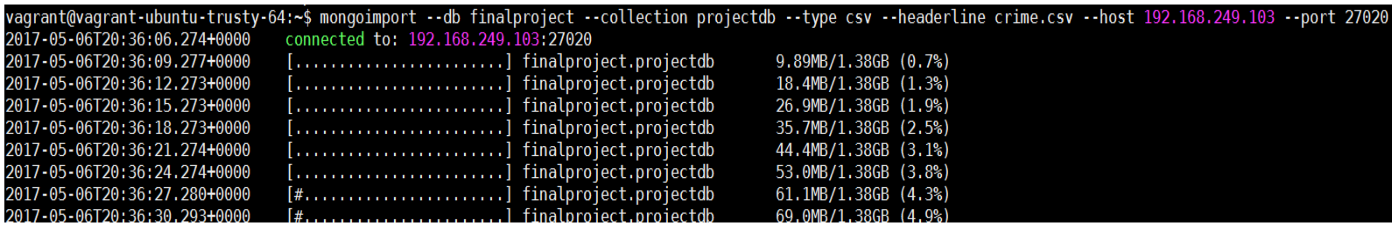
```
mongos> sh.enableSharding("finalproject")
{ "ok" : 1 }
mongos> sh.shardCollection("finalproject.projectdb",{Year:1})
{ "collectionsharded" : "finalproject.projectdb", "ok" : 1 }
```

xviii. Status after enabling sharding.



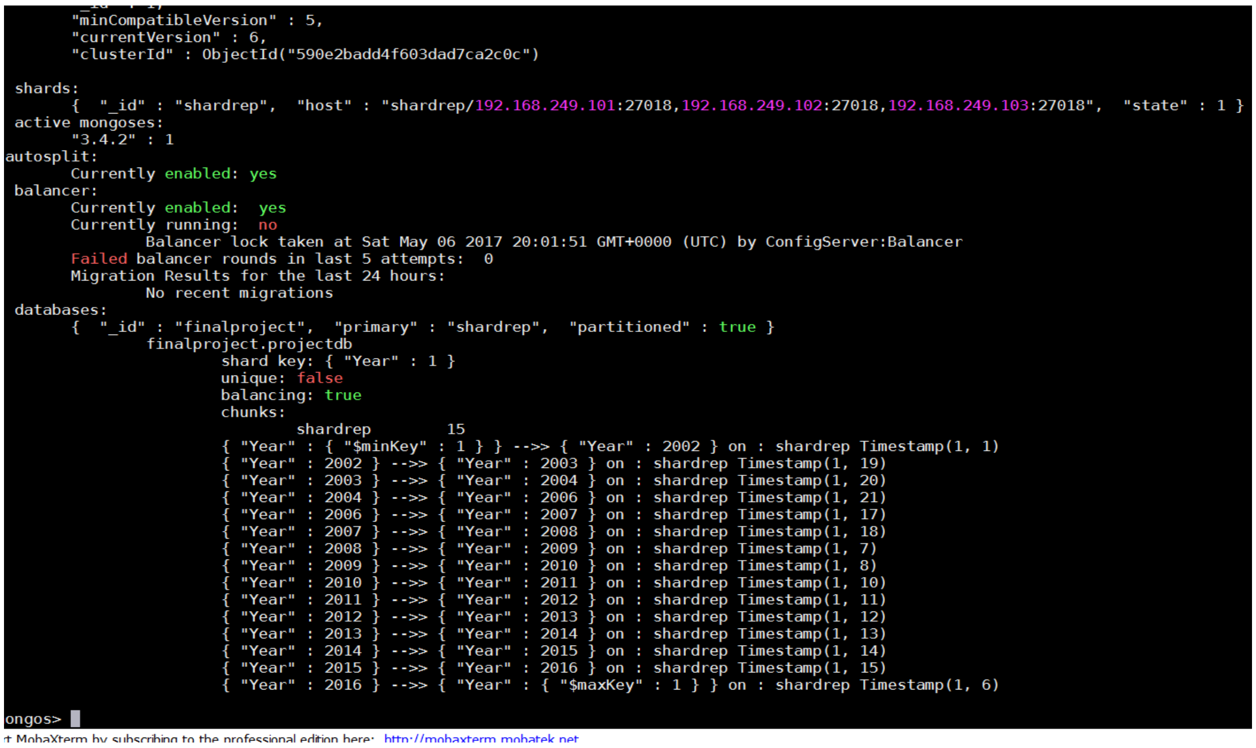
```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("590e2badd4f603dad7ca2c0c")
}
  shards:
        {  "_id" : "shardrep",  "host" : "shardrep/192.168.249.101:27018,192.168.249.102:27018,192.168.249.103:27018",  "state" :
  active mongoses:
        "3.4.2" : 1
 autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Sat May 06 2017 20:01:51 GMT+0000 (UTC) by ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "finalproject",  "primary" : "shardrep",  "partitioned" : true }
                finalproject.projectdb
                        shard key: { "Year" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shardrep        1
                        { "Year" : { "$minKey" : 1 } } -->> { "Year" : { "$maxKey" : 1 } } on : shardrep Timestamp(1, 0)
```

xix. Importing data on mongos.



```
vagrant@vagrant-ubuntu-trusty-64:~$ mongoimport --db finalproject --collection projectdb --type csv --headerline crime.csv --host 192.168.249.103 --port 27020
2017-05-06T20:36:06.274+0000    connected to: 192.168.249.103:27020
2017-05-06T20:36:09.277+0000    [........................] finalproject.projectdb      9.89MB/1.38GB (0.7%)
2017-05-06T20:36:12.273+0000    [........................] finalproject.projectdb      18.4MB/1.38GB (1.3%)
2017-05-06T20:36:15.273+0000    [........................] finalproject.projectdb      26.9MB/1.38GB (1.9%)
2017-05-06T20:36:18.273+0000    [........................] finalproject.projectdb      35.7MB/1.38GB (2.5%)
2017-05-06T20:36:21.274+0000    [........................] finalproject.projectdb      44.4MB/1.38GB (3.1%)
2017-05-06T20:36:24.274+0000    [........................] finalproject.projectdb      53.0MB/1.38GB (3.8%)
2017-05-06T20:36:27.280+0000    [#.......................] finalproject.projectdb      61.1MB/1.38GB (4.3%)
2017-05-06T20:36:30.293+0000    [#.......................] finalproject.projectdb      69.0MB/1.38GB (4.9%)
```

xx. Status after importing data.



```
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("590e2badd4f603dad7ca2c0c")

 shards:
        {  "_id" : "shardrep",  "host" : "shardrep/192.168.249.101:27018,192.168.249.102:27018,192.168.249.103:27018",  "state" : 1 }
 active mongoses:
        "3.4.2" : 1
autosplit:
        Currently enabled: yes
 balancer:
        Currently enabled:  yes
        Currently running:  no
                Balancer lock taken at Sat May 06 2017 20:01:51 GMT+0000 (UTC) by ConfigServer:Balancer
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
 databases:
        {  "_id" : "finalproject",  "primary" : "shardrep",  "partitioned" : true }
                finalproject.projectdb
                        shard key: { "Year" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                shardrep        15
                        { "Year" : { "$minKey" : 1 } } -->> { "Year" : 2002 } on : shardrep Timestamp(1, 1)
                        { "Year" : 2002 } -->> { "Year" : 2003 } on : shardrep Timestamp(1, 19)
                        { "Year" : 2003 } -->> { "Year" : 2004 } on : shardrep Timestamp(1, 20)
                        { "Year" : 2004 } -->> { "Year" : 2006 } on : shardrep Timestamp(1, 21)
                        { "Year" : 2006 } -->> { "Year" : 2007 } on : shardrep Timestamp(1, 17)
                        { "Year" : 2007 } -->> { "Year" : 2008 } on : shardrep Timestamp(1, 18)
                        { "Year" : 2008 } -->> { "Year" : 2009 } on : shardrep Timestamp(1, 7)
                        { "Year" : 2009 } -->> { "Year" : 2010 } on : shardrep Timestamp(1, 8)
                        { "Year" : 2010 } -->> { "Year" : 2011 } on : shardrep Timestamp(1, 10)
                        { "Year" : 2011 } -->> { "Year" : 2012 } on : shardrep Timestamp(1, 11)
                        { "Year" : 2012 } -->> { "Year" : 2013 } on : shardrep Timestamp(1, 12)
                        { "Year" : 2013 } -->> { "Year" : 2014 } on : shardrep Timestamp(1, 13)
                        { "Year" : 2014 } -->> { "Year" : 2015 } on : shardrep Timestamp(1, 14)
                        { "Year" : 2015 } -->> { "Year" : 2016 } on : shardrep Timestamp(1, 15)
                        { "Year" : 2016 } -->> { "Year" : { "$maxKey" : 1 } } on : shardrep Timestamp(1, 6)

ongos>
t MobaXterm by subscribing to the professional edition here:  http://mobaxterm.mobatek.net
```

10. NoSQL Driver

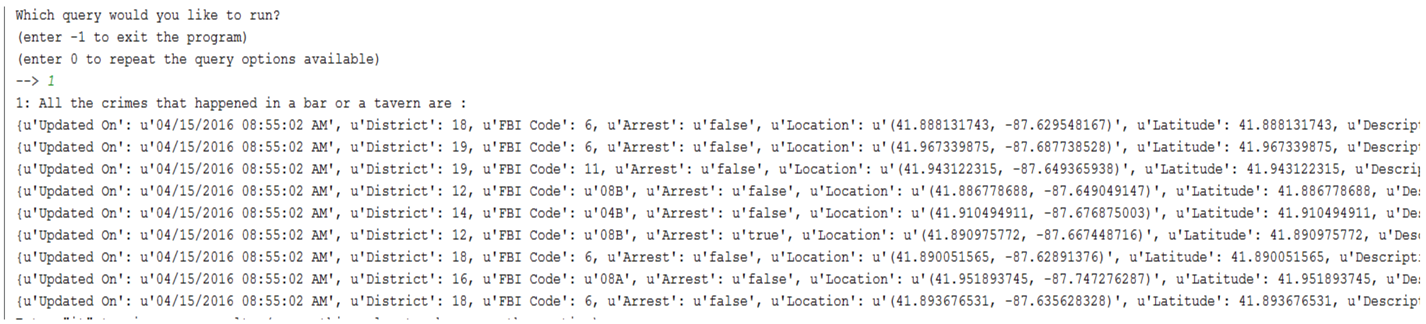
i. PyMongo Driver

- The PyMongo distribution contains tools for interacting with MongoDB database from Python. The bson package is an implementation of the BSON format for Python. The pymongo package is a native Python driver for MongoDB.
- For installation, we downloaded the package and ran the python setup.py install command.
- The reason we chose PyMongo is because it is the most efficient tool for working with MongoDB using the utilities of Python language.
- At the core of PyMongo is the MongoClient object, which is used to make connections and queries to a MongoDB database cluster. It can be used to connect to a standalone mongod instance, a replica set or mongos instances.

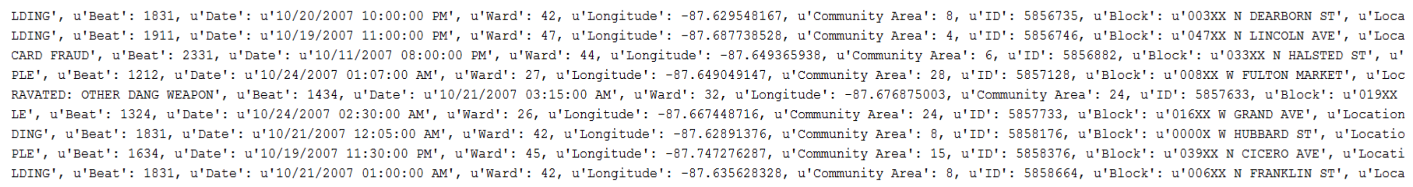
ii. Python Application to Database

- The system is connected to mongos using the host and port number specified above using MongoClient function.
- The function is provided with a time limit in which it tries to establish a connection otherwise gives an error message and program ends.
- After connection is established the program is feeded the database name and then the collection name is specified.

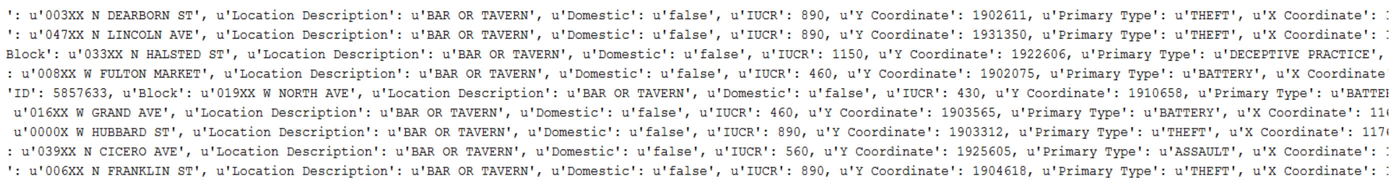
- Since the connection between the application and the MongoDB is established we can start querying the data.
- A list of all the available queries is printed on the screen using Python print command. Every query has a serial number associated with it. The input given by the user is the serial number.
- If the user gives wrong input, then the program prints the error message and the options are repeated.
- Based on the input from user, corresponding function is called. The database returns queries in the form of rows. The function involves the name of the connection, along with database name and collection and the query that is to be executed. Every serial number has its corresponding query.
- The data that is sent from MongoDb is printed till the data is over or user chooses to end the query. If there is no data left to print, the system returns '-1' which means to end the query.
- After the execution is completed and use decides to close the application, the established connection is closed using the close function.
- Screenshot from Python program printing the crimes happened in a bar or tavern.
-



```
Which query would you like to run?
(enter -1 to exit the program)
(enter 0 to repeat the query options available)
--> 1
1: All the crimes that happened in a bar or a tavern are :
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 18, u'FBI Code': 6, u'Arrest': u'false', u'Location': u'(41.888131743, -87.629548167)', u'Latitude': 41.888131743, u'Descrip
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 19, u'FBI Code': 6, u'Arrest': u'false', u'Location': u'(41.967339875, -87.687738528)', u'Latitude': 41.967339875, u'Descrip
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 19, u'FBI Code': 11, u'Arrest': u'false', u'Location': u'(41.943122315, -87.649365938)', u'Latitude': 41.943122315, u'Descri
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 12, u'FBI Code': u'08B', u'Arrest': u'false', u'Location': u'(41.886778688, -87.649049147)', u'Latitude': 41.886778688, u'De
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 14, u'FBI Code': u'04B', u'Arrest': u'false', u'Location': u'(41.910494911, -87.676875003)', u'Latitude': 41.910494911, u'De
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 12, u'FBI Code': u'08B', u'Arrest': u'true', u'Location': u'(41.890975772, -87.667448716)', u'Latitude': 41.890975772, u'Des
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 18, u'FBI Code': 6, u'Arrest': u'false', u'Location': u'(41.890051565, -87.62891376)', u'Latitude': 41.890051565, u'Descript
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 16, u'FBI Code': u'08A', u'Arrest': u'false', u'Location': u'(41.951893745, -87.747276287)', u'Latitude': 41.951893745, u'De
{u'Updated On': u'04/15/2016 08:55:02 AM', u'District': 18, u'FBI Code': 6, u'Arrest': u'false', u'Location': u'(41.893676531, -87.635628328)', u'Latitude': 41.893676531, u'Descrip
```



```
LDING', u'Beat': 1831, u'Date': u'10/20/2007 10:00:00 PM', u'Ward': 42, u'Longitude': -87.629548167, u'Community Area': 8, u'ID': 5856735, u'Block': u'003XX N DEARBORN ST', u'Loca
LDING', u'Beat': 1911, u'Date': u'10/19/2007 11:00:00 PM', u'Ward': 47, u'Longitude': -87.687738528, u'Community Area': 4, u'ID': 5856746, u'Block': u'047XX N LINCOLN AVE', u'Loca
CARD FRAUD', u'Beat': 2331, u'Date': u'10/11/2007 08:00:00 PM', u'Ward': 44, u'Longitude': -87.649365938, u'Community Area': 6, u'ID': 5856882, u'Block': u'033XX N HALSTED ST', u'
PLE', u'Beat': 1212, u'Date': u'10/24/2007 01:07:00 AM', u'Ward': 27, u'Longitude': -87.649049147, u'Community Area': 28, u'ID': 5857128, u'Block': u'008XX W FULTON MARKET', u'Loc
RAVATED: OTHER DANG WEAPON', u'Beat': 1434, u'Date': u'10/21/2007 03:15:00 AM', u'Ward': 32, u'Longitude': -87.676875003, u'Community Area': 24, u'ID': 5857633, u'Block': u'019XX
LE', u'Beat': 1324, u'Date': u'10/24/2007 02:30:00 AM', u'Ward': 26, u'Longitude': -87.667448716, u'Community Area': 24, u'ID': 5857733, u'Block': u'016XX W GRAND AVE', u'Location
DING', u'Beat': 1831, u'Date': u'10/21/2007 12:05:00 AM', u'Ward': 42, u'Longitude': -87.62891376, u'Community Area': 8, u'ID': 5858176, u'Block': u'0000X W HUBBARD ST', u'Locatio
PLE', u'Beat': 1634, u'Date': u'10/19/2007 11:30:00 PM', u'Ward': 45, u'Longitude': -87.747276287, u'Community Area': 15, u'ID': 5858376, u'Block': u'039XX N CICERO AVE', u'Locati
LDING', u'Beat': 1831, u'Date': u'10/21/2007 01:00:00 AM', u'Ward': 42, u'Longitude': -87.635628328, u'Community Area': 8, u'ID': 5858664, u'Block': u'006XX N FRANKLIN ST', u'Loca
```



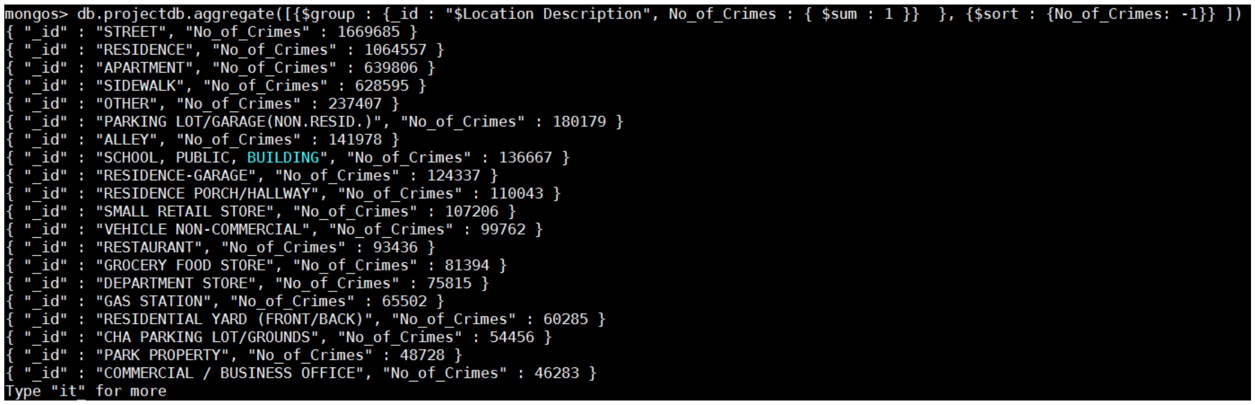
```
': u'003XX N DEARBORN ST', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 890, u'Y Coordinate': 1902611, u'Primary Type': u'THEFT', u'X Coordinate':
': u'047XX N LINCOLN AVE', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 890, u'Y Coordinate': 1931350, u'Primary Type': u'THEFT', u'X Coordinate':
Block': u'033XX N HALSTED ST', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 1150, u'Y Coordinate': 1922606, u'Primary Type': u'DECEPTIVE PRACTICE',
: u'008XX W FULTON MARKET', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 460, u'Y Coordinate': 1902075, u'Primary Type': u'BATTERY', u'X Coordinate
'ID': 5857633, u'Block': u'019XX W NORTH AVE', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 430, u'Y Coordinate': 1910658, u'Primary Type': u'BATTE
 u'016XX W GRAND AVE', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 460, u'Y Coordinate': 1903565, u'Primary Type': u'BATTERY', u'X Coordinate': 11
 u'0000X W HUBBARD ST', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 890, u'Y Coordinate': 1903312, u'Primary Type': u'THEFT', u'X Coordinate': 117
: u'039XX N CICERO AVE', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 560, u'Y Coordinate': 1925605, u'Primary Type': u'ASSAULT', u'X Coordinate':
': u'006XX N FRANKLIN ST', u'Location Description': u'BAR OR TAVERN', u'Domestic': u'false', u'IUCR': 890, u'Y Coordinate': 1904618, u'Primary Type': u'THEFT', u'X Coordinate':
```

11. Use Cases

Use Case 1

Description: Where does the crime happen most often? (Street, Home, etc)'

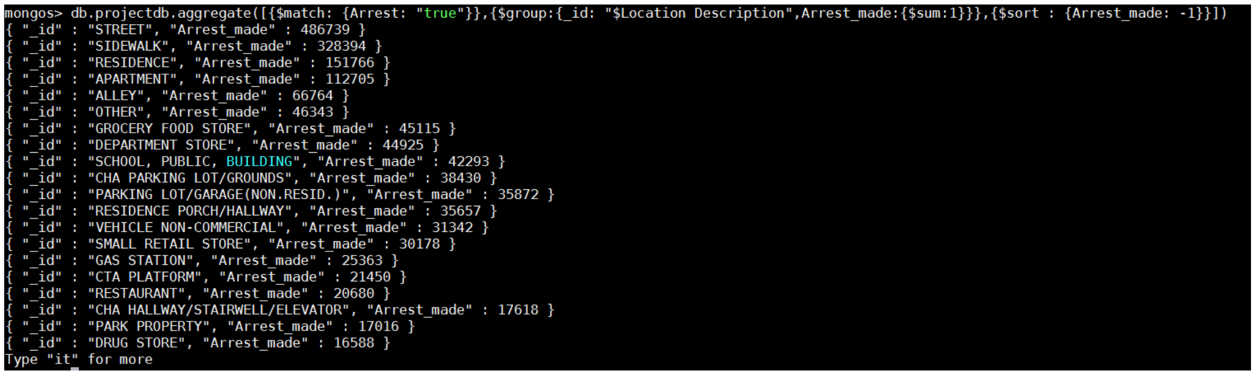
Use Case: db.projectdb.aggregate([{ $group : { _id : "$loc_desc", No_of_Crimes : { $sum : 1 } } }, { $sort : { No_of_Crimes: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$group : {_id : "$Location Description", No_of_Crimes : { $sum : 1 }}  }, {$sort : {No_of_Crimes: -1}} ])
{ "_id" : "STREET", "No_of_Crimes" : 1669685 }
{ "_id" : "RESIDENCE", "No_of_Crimes" : 1064557 }
{ "_id" : "APARTMENT", "No_of_Crimes" : 639806 }
{ "_id" : "SIDEWALK", "No_of_Crimes" : 628595 }
{ "_id" : "OTHER", "No_of_Crimes" : 237407 }
{ "_id" : "PARKING LOT/GARAGE(NON.RESID.)", "No_of_Crimes" : 180179 }
{ "_id" : "ALLEY", "No_of_Crimes" : 141978 }
{ "_id" : "SCHOOL, PUBLIC, BUILDING", "No_of_Crimes" : 136667 }
{ "_id" : "RESIDENCE-GARAGE", "No_of_Crimes" : 124337 }
{ "_id" : "RESIDENCE PORCH/HALLWAY", "No_of_Crimes" : 110043 }
{ "_id" : "SMALL RETAIL STORE", "No_of_Crimes" : 107206 }
{ "_id" : "VEHICLE NON-COMMERCIAL", "No_of_Crimes" : 99762 }
{ "_id" : "RESTAURANT", "No_of_Crimes" : 93436 }
{ "_id" : "GROCERY FOOD STORE", "No_of_Crimes" : 81394 }
{ "_id" : "DEPARTMENT STORE", "No_of_Crimes" : 75815 }
{ "_id" : "GAS STATION", "No_of_Crimes" : 65502 }
{ "_id" : "RESIDENTIAL YARD (FRONT/BACK)", "No_of_Crimes" : 60285 }
{ "_id" : "CHA PARKING LOT/GROUNDS", "No_of_Crimes" : 54456 }
{ "_id" : "PARK PROPERTY", "No_of_Crimes" : 48728 }
{ "_id" : "COMMERCIAL / BUSINESS OFFICE", "No_of_Crimes" : 46283 }
Type "it" for more
```

Use Case 2

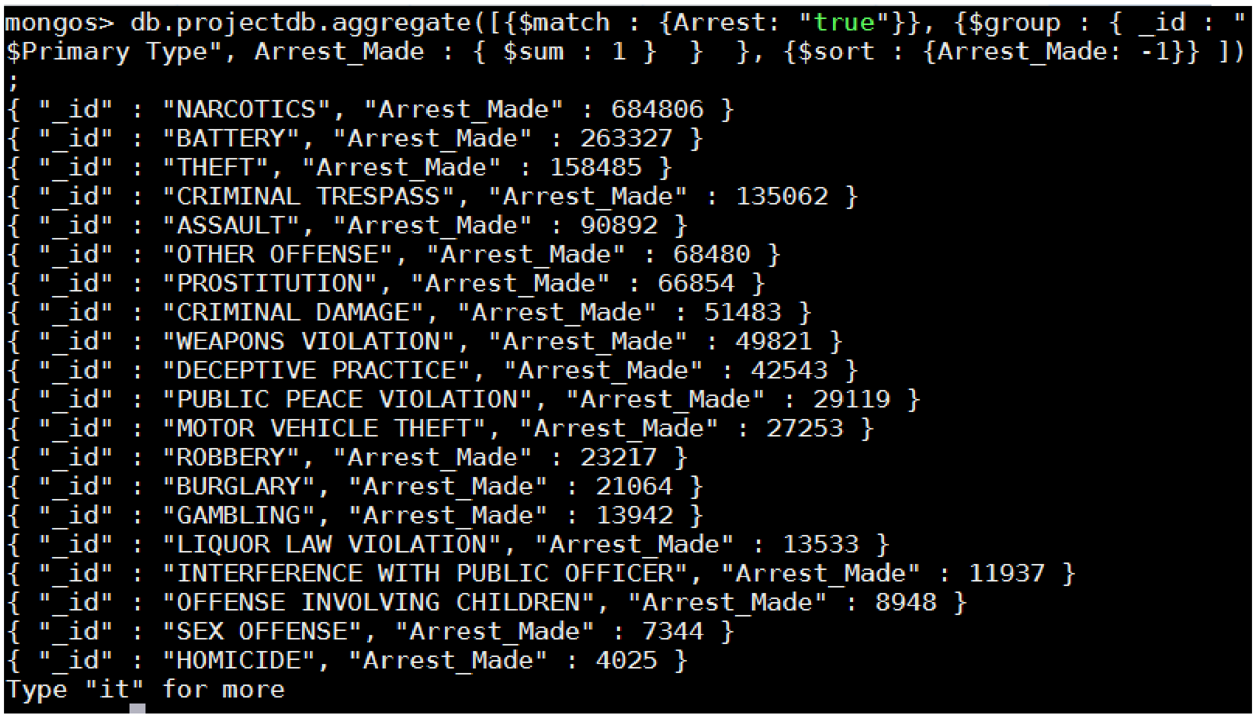
Description: Max crime location wherein the police could make an arrest
Query: db.projectdb.aggregate([{ $match: { Arrest: "true" } }, { $group: { _id: "$Location Description", Arrest_made: { $sum: 1 } } }, { $sort : { Arrest_made: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {Arrest: "true"}},{$group:{_id: "$Location Description",Arrest_made:{$sum:1}}},{$sort : {Arrest_made: -1}}])
{ "_id" : "STREET", "Arrest_made" : 486739 }
{ "_id" : "SIDEWALK", "Arrest_made" : 328394 }
{ "_id" : "RESIDENCE", "Arrest_made" : 151766 }
{ "_id" : "APARTMENT", "Arrest_made" : 112705 }
{ "_id" : "ALLEY", "Arrest_made" : 66764 }
{ "_id" : "OTHER", "Arrest_made" : 46343 }
{ "_id" : "GROCERY FOOD STORE", "Arrest_made" : 45115 }
{ "_id" : "DEPARTMENT STORE", "Arrest_made" : 44925 }
{ "_id" : "SCHOOL, PUBLIC, BUILDING", "Arrest_made" : 42293 }
{ "_id" : "CHA PARKING LOT/GROUNDS", "Arrest_made" : 38430 }
{ "_id" : "PARKING LOT/GARAGE(NON.RESID.)", "Arrest_made" : 35872 }
{ "_id" : "RESIDENCE PORCH/HALLWAY", "Arrest_made" : 35657 }
{ "_id" : "VEHICLE NON-COMMERCIAL", "Arrest_made" : 31342 }
{ "_id" : "SMALL RETAIL STORE", "Arrest_made" : 30178 }
{ "_id" : "GAS STATION", "Arrest_made" : 25363 }
{ "_id" : "CTA PLATFORM", "Arrest_made" : 21450 }
{ "_id" : "RESTAURANT", "Arrest_made" : 20680 }
{ "_id" : "CHA HALLWAY/STAIRWELL/ELEVATOR", "Arrest_made" : 17618 }
{ "_id" : "PARK PROPERTY", "Arrest_made" : 17016 }
{ "_id" : "DRUG STORE", "Arrest_made" : 16588 }
Type "it" for more
```

Use Case 3

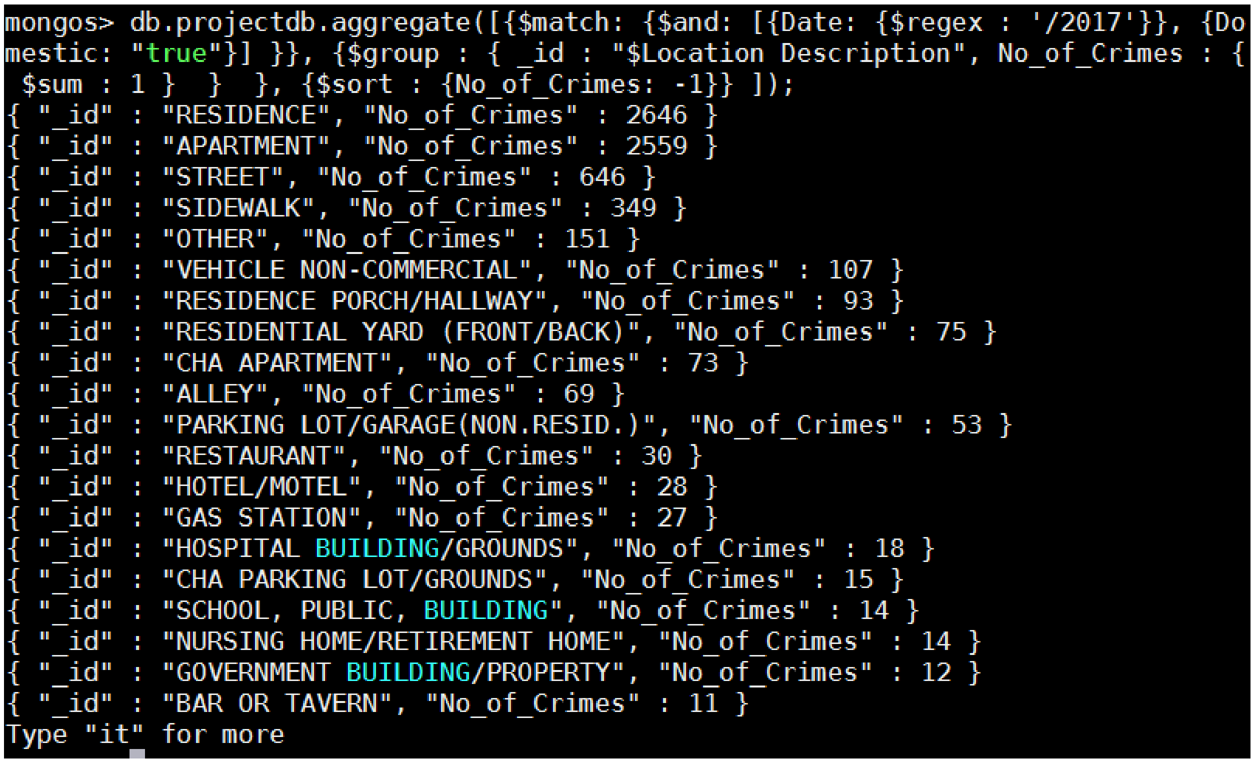
Description: Crime in which most arrests were made
Query: db.projectdb.aggregate([{ $match : { Arrest: "true" } }, { $group : { _id : "$Primary Type", Arrest_Made : { $sum : 1 } } }, { $sort : { Arrest_Made: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match : {Arrest: "true"}}, {$group : { _id : "
$Primary Type", Arrest_Made : { $sum : 1 }  }  }, {$sort : {Arrest_Made: -1}} ])
;
{ "_id" : "NARCOTICS", "Arrest_Made" : 684806 }
{ "_id" : "BATTERY", "Arrest_Made" : 263327 }
{ "_id" : "THEFT", "Arrest_Made" : 158485 }
{ "_id" : "CRIMINAL TRESPASS", "Arrest_Made" : 135062 }
{ "_id" : "ASSAULT", "Arrest_Made" : 90892 }
{ "_id" : "OTHER OFFENSE", "Arrest_Made" : 68480 }
{ "_id" : "PROSTITUTION", "Arrest_Made" : 66854 }
{ "_id" : "CRIMINAL DAMAGE", "Arrest_Made" : 51483 }
{ "_id" : "WEAPONS VIOLATION", "Arrest_Made" : 49821 }
{ "_id" : "DECEPTIVE PRACTICE", "Arrest_Made" : 42543 }
{ "_id" : "PUBLIC PEACE VIOLATION", "Arrest_Made" : 29119 }
{ "_id" : "MOTOR VEHICLE THEFT", "Arrest_Made" : 27253 }
{ "_id" : "ROBBERY", "Arrest_Made" : 23217 }
{ "_id" : "BURGLARY", "Arrest_Made" : 21064 }
{ "_id" : "GAMBLING", "Arrest_Made" : 13942 }
{ "_id" : "LIQUOR LAW VIOLATION", "Arrest_Made" : 13533 }
{ "_id" : "INTERFERENCE WITH PUBLIC OFFICER", "Arrest_Made" : 11937 }
{ "_id" : "OFFENSE INVOLVING CHILDREN", "Arrest_Made" : 8948 }
{ "_id" : "SEX OFFENSE", "Arrest_Made" : 7344 }
{ "_id" : "HOMICIDE", "Arrest_Made" : 4025 }
Type "it" for more
```

Use Case 4

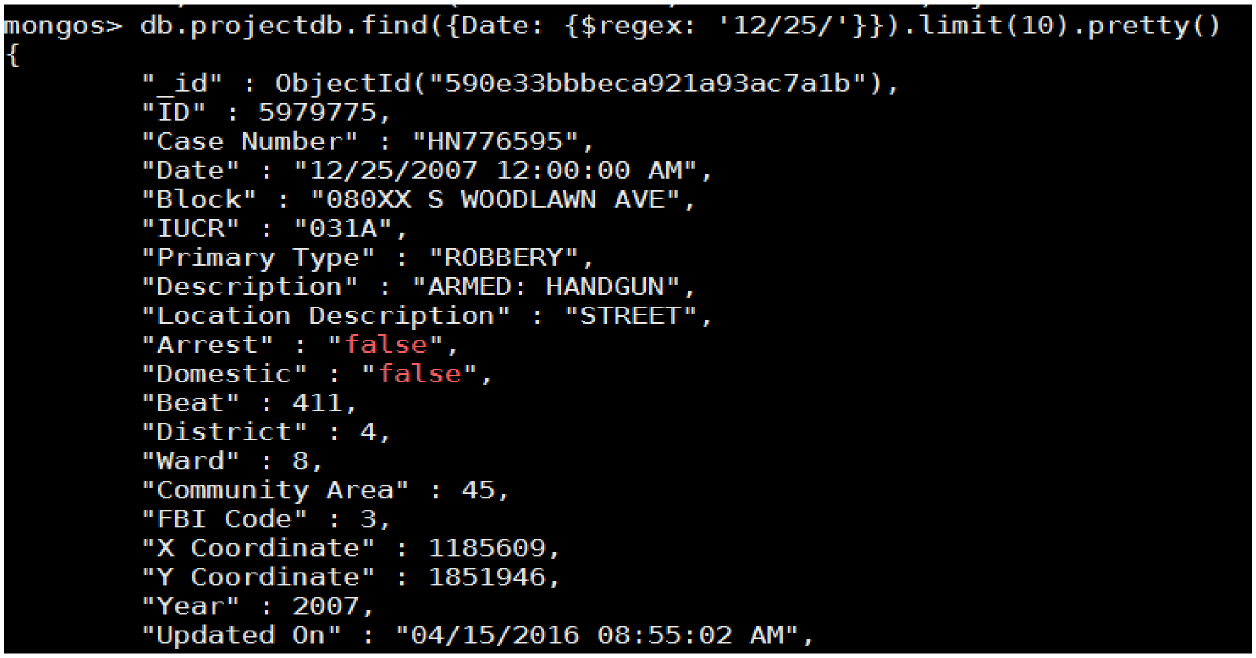
Description: Number of Domestic violence cases in 2017 and the location where it occurred.
Query: db.projectdb.aggregate([{ $match: { $and: [{ Date: { $regex : '/2017' } }, { Domestic: "true" }] } }, { $group : { _id : "$Location Description", No_of_Crimes: { $sum: 1 } } }, { $sort: { No_of_Crimes: -1 } }])
Output:



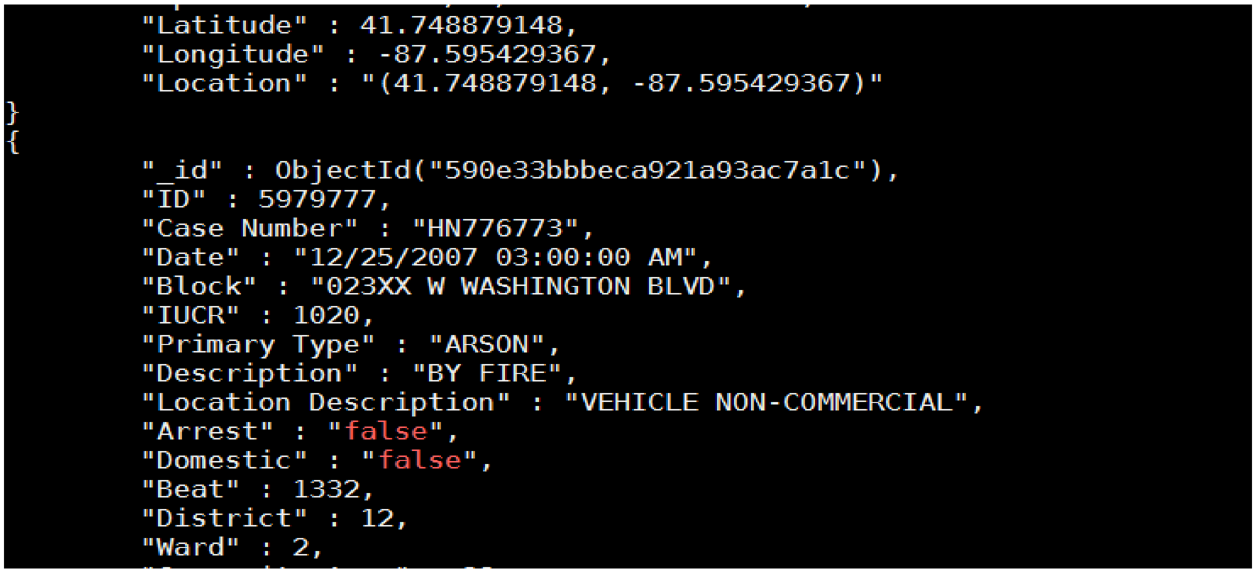
```
mongos> db.projectdb.aggregate([{$match: {$and: [{Date: {$regex : '/2017'}}, {Do
mestic: "true"}] }}, {$group : { _id : "$Location Description", No_of_Crimes : {
 $sum : 1 }  }  }, {$sort : {No_of_Crimes: -1}} ]);
{ "_id" : "RESIDENCE", "No_of_Crimes" : 2646 }
{ "_id" : "APARTMENT", "No_of_Crimes" : 2559 }
{ "_id" : "STREET", "No_of_Crimes" : 646 }
{ "_id" : "SIDEWALK", "No_of_Crimes" : 349 }
{ "_id" : "OTHER", "No_of_Crimes" : 151 }
{ "_id" : "VEHICLE NON-COMMERCIAL", "No_of_Crimes" : 107 }
{ "_id" : "RESIDENCE PORCH/HALLWAY", "No_of_Crimes" : 93 }
{ "_id" : "RESIDENTIAL YARD (FRONT/BACK)", "No_of_Crimes" : 75 }
{ "_id" : "CHA APARTMENT", "No_of_Crimes" : 73 }
{ "_id" : "ALLEY", "No_of_Crimes" : 69 }
{ "_id" : "PARKING LOT/GARAGE(NON.RESID.)", "No_of_Crimes" : 53 }
{ "_id" : "RESTAURANT", "No_of_Crimes" : 30 }
{ "_id" : "HOTEL/MOTEL", "No_of_Crimes" : 28 }
{ "_id" : "GAS STATION", "No_of_Crimes" : 27 }
{ "_id" : "HOSPITAL BUILDING/GROUNDS", "No_of_Crimes" : 18 }
{ "_id" : "CHA PARKING LOT/GROUNDS", "No_of_Crimes" : 15 }
{ "_id" : "SCHOOL, PUBLIC, BUILDING", "No_of_Crimes" : 14 }
{ "_id" : "NURSING HOME/RETIREMENT HOME", "No_of_Crimes" : 14 }
{ "_id" : "GOVERNMENT BUILDING/PROPERTY", "No_of_Crimes" : 12 }
{ "_id" : "BAR OR TAVERN", "No_of_Crimes" : 11 }
Type "it" for more
```

Use Case 5:

Description: List of crime records where crime took place on Dec 25:
Query: db.projectdb.find({ Date: { $regex: '12/25/' } }).limit(10).pretty()
Output:



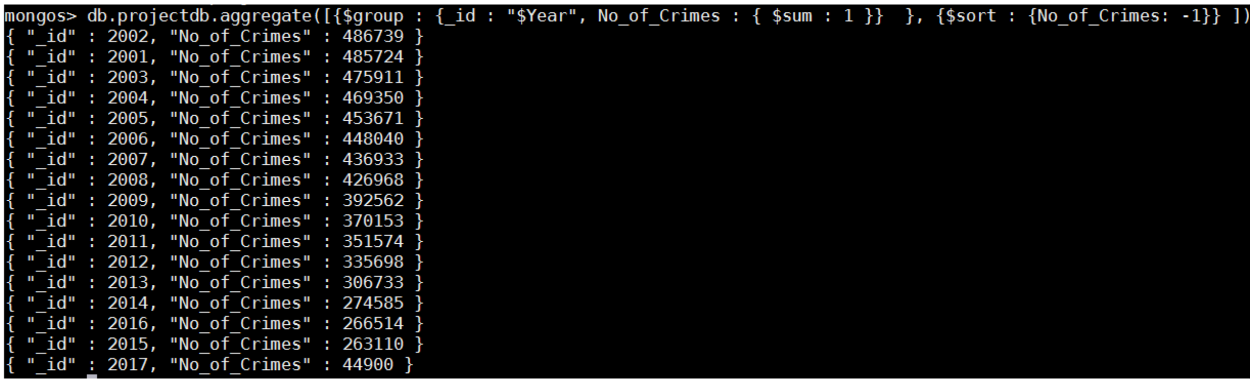
```
mongos> db.projectdb.find({Date: {$regex: '12/25/'}}).limit(10).pretty()
{
        "_id" : ObjectId("590e33bbbeca921a93ac7a1b"),
        "ID" : 5979775,
        "Case Number" : "HN776595",
        "Date" : "12/25/2007 12:00:00 AM",
        "Block" : "080XX S WOODLAWN AVE",
        "IUCR" : "031A",
        "Primary Type" : "ROBBERY",
        "Description" : "ARMED: HANDGUN",
        "Location Description" : "STREET",
        "Arrest" : "false",
        "Domestic" : "false",
        "Beat" : 411,
        "District" : 4,
        "Ward" : 8,
        "Community Area" : 45,
        "FBI Code" : 3,
        "X Coordinate" : 1185609,
        "Y Coordinate" : 1851946,
        "Year" : 2007,
        "Updated On" : "04/15/2016 08:55:02 AM",
```



```
        "Latitude" : 41.748879148,
        "Longitude" : -87.595429367,
        "Location" : "(41.748879148, -87.595429367)"
}
{
        "_id" : ObjectId("590e33bbbeca921a93ac7a1c"),
        "ID" : 5979777,
        "Case Number" : "HN776773",
        "Date" : "12/25/2007 03:00:00 AM",
        "Block" : "023XX W WASHINGTON BLVD",
        "IUCR" : 1020,
        "Primary Type" : "ARSON",
        "Description" : "BY FIRE",
        "Location Description" : "VEHICLE NON-COMMERCIAL",
        "Arrest" : "false",
        "Domestic" : "false",
        "Beat" : 1332,
        "District" : 12,
        "Ward" : 2,
```

Use Case 6

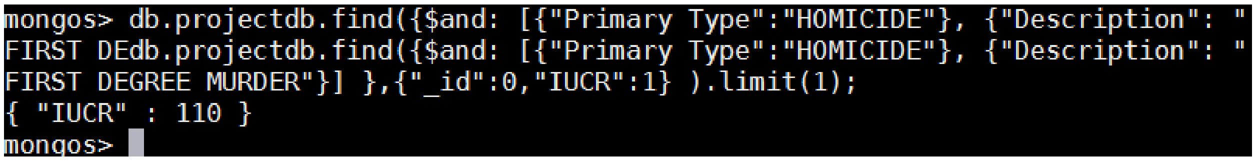
Description: Which year saw the maximum number of crimes in Chicago? Also, what are the number of crimes that happened in Chicago categorized by year?
Query: db.projectdb.aggregate([{ $group: { _id: "$Year", No_of_Crimes: { $sum: 1 } } }, { $sort: { No_of_Crimes: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$group : {_id : "$Year", No_of_Crimes : { $sum : 1 }}  }, {$sort : {No_of_Crimes: -1}} ])
{ "_id" : 2002, "No_of_Crimes" : 486739 }
{ "_id" : 2001, "No_of_Crimes" : 485724 }
{ "_id" : 2003, "No_of_Crimes" : 475911 }
{ "_id" : 2004, "No_of_Crimes" : 469350 }
{ "_id" : 2005, "No_of_Crimes" : 453671 }
{ "_id" : 2006, "No_of_Crimes" : 448040 }
{ "_id" : 2007, "No_of_Crimes" : 436933 }
{ "_id" : 2008, "No_of_Crimes" : 426968 }
{ "_id" : 2009, "No_of_Crimes" : 392562 }
{ "_id" : 2010, "No_of_Crimes" : 370153 }
{ "_id" : 2011, "No_of_Crimes" : 351574 }
{ "_id" : 2012, "No_of_Crimes" : 335698 }
{ "_id" : 2013, "No_of_Crimes" : 306733 }
{ "_id" : 2014, "No_of_Crimes" : 274585 }
{ "_id" : 2016, "No_of_Crimes" : 266514 }
{ "_id" : 2015, "No_of_Crimes" : 263110 }
{ "_id" : 2017, "No_of_Crimes" : 44900 }
```

Use Case 7

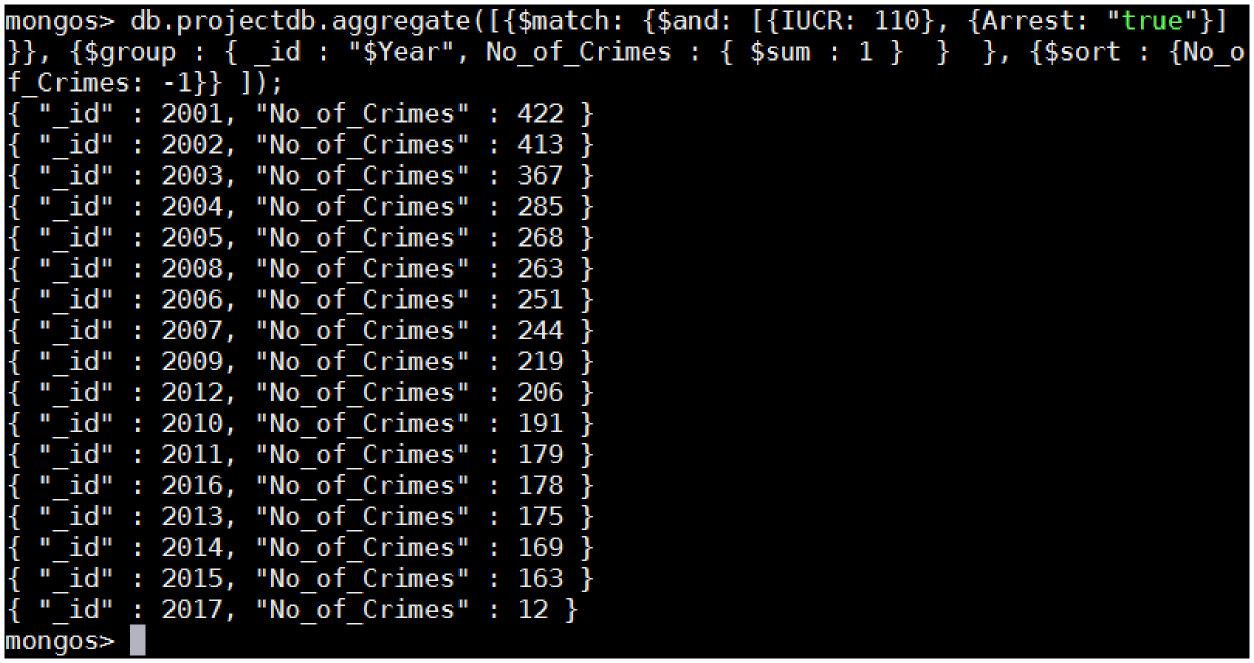
Description: What is the IUCR code for the crime where the Primary Type is Homicide and Secondary Type is First Degree Murder. This IUCR code can be used to locate the number of Homicide- First degree murders per year.
Query: db.projectdb.find({ $and: [{ "Primary Type": "HOMICIDE" }, { "Description": "FIRST DEGREE MURDER" }] }, { "_id": 0, "IUCR": 1 }).limit(1)
Output:



```
mongos> db.projectdb.find({$and: [{"Primary Type":"HOMICIDE"}, {"Description": "
FIRST DEdb.projectdb.find({$and: [{"Primary Type":"HOMICIDE"}, {"Description": "
FIRST DEGREE MURDER"}] },{"_id":0,"IUCR":1} ).limit(1);
{ "IUCR" : 110 }
mongos>
```

Use Case 8

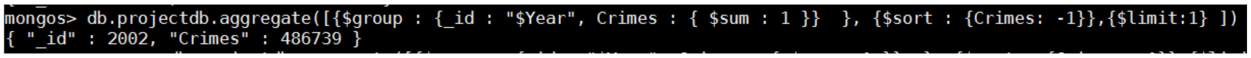
Description: How many First Degree Homicide murders (based on the IUCR code that we found in the previous step) were reported in which an arrest was made, per year.
Query: db.projectdb.aggregate([{ $match: { $and: [{ IUCR: 110 }, { Arrest: "true" }] } }, { $group: { _id: "$Year", No_of_Crimes: { $sum: 1 } } }, { $sort: { No_of_Crimes: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {$and: [{IUCR: 110}, {Arrest: "true"}]
}}, {$group : { _id : "$Year", No_of_Crimes : { $sum : 1 }  }  }, {$sort : {No_o
f_Crimes: -1}} ]);
{ "_id" : 2001, "No_of_Crimes" : 422 }
{ "_id" : 2002, "No_of_Crimes" : 413 }
{ "_id" : 2003, "No_of_Crimes" : 367 }
{ "_id" : 2004, "No_of_Crimes" : 285 }
{ "_id" : 2005, "No_of_Crimes" : 268 }
{ "_id" : 2008, "No_of_Crimes" : 263 }
{ "_id" : 2006, "No_of_Crimes" : 251 }
{ "_id" : 2007, "No_of_Crimes" : 244 }
{ "_id" : 2009, "No_of_Crimes" : 219 }
{ "_id" : 2012, "No_of_Crimes" : 206 }
{ "_id" : 2010, "No_of_Crimes" : 191 }
{ "_id" : 2011, "No_of_Crimes" : 179 }
{ "_id" : 2016, "No_of_Crimes" : 178 }
{ "_id" : 2013, "No_of_Crimes" : 175 }
{ "_id" : 2014, "No_of_Crimes" : 169 }
{ "_id" : 2015, "No_of_Crimes" : 163 }
{ "_id" : 2017, "No_of_Crimes" : 12 }
mongos>
```

Use Case 9

Description: Year with the most number of crimes
Query: db.projectdb.aggregate([{ $group: { _id: "$Year", Crimes: { $sum: 1 } } }, { $sort: { Crimes: -1 } }, { $limit:1 }])
Output:

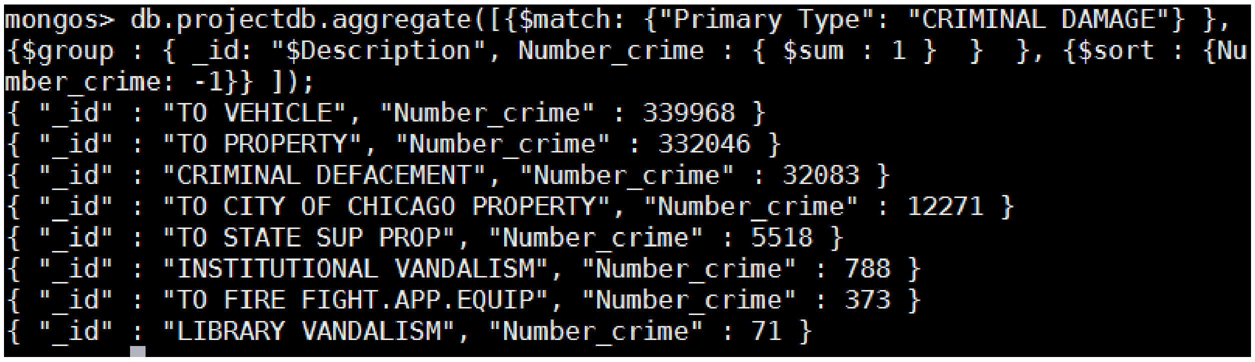


```
mongos> db.projectdb.aggregate([{$group : {_id : "$Year", Crimes : { $sum : 1 }}  }, {$sort : {Crimes: -1}},{$limit:1} ])
{ "_id" : 2002, "Crimes" : 486739 }
```

Use Case 10

Description: The most common type of secondary crime in descending order with the primary crime category specified ("CRIMINAL DAMAGE")

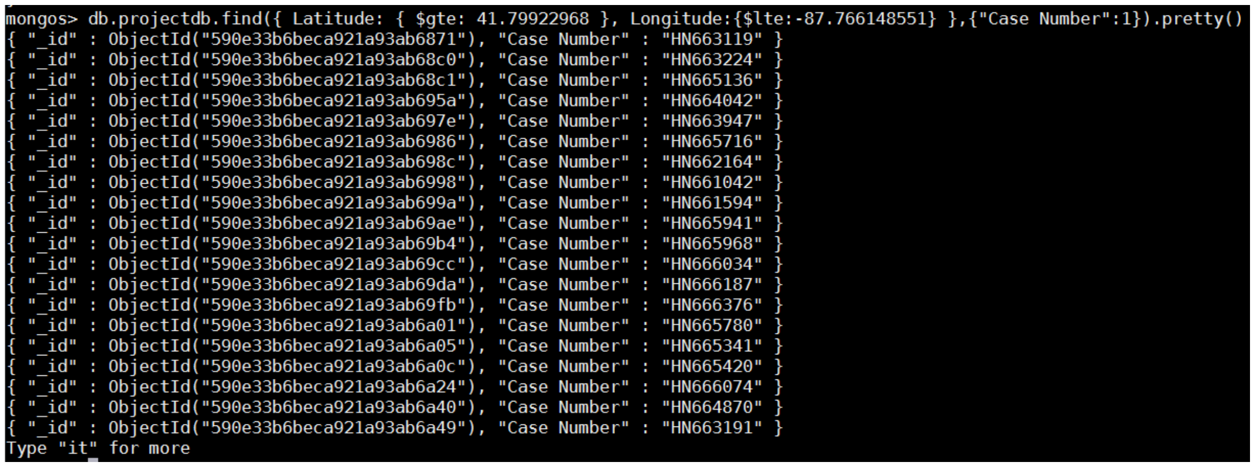
Query: db.projectdb.aggregate([{ $match: { "Primary Type": "CRIMINAL DAMAGE" } }, { $group: { _id: "$Description", Number_crime: { $sum: 1 } } }, { $sort: { Number_crime: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {"Primary Type": "CRIMINAL DAMAGE"} },
{$group : { _id: "$Description", Number_crime : { $sum : 1 }  }  }, {$sort : {Nu
mber_crime: -1}} ]);
{ "_id" : "TO VEHICLE", "Number_crime" : 339968 }
{ "_id" : "TO PROPERTY", "Number_crime" : 332046 }
{ "_id" : "CRIMINAL DEFACEMENT", "Number_crime" : 32083 }
{ "_id" : "TO CITY OF CHICAGO PROPERTY", "Number_crime" : 12271 }
{ "_id" : "TO STATE SUP PROP", "Number_crime" : 5518 }
{ "_id" : "INSTITUTIONAL VANDALISM", "Number_crime" : 788 }
{ "_id" : "TO FIRE FIGHT.APP.EQUIP", "Number_crime" : 373 }
{ "_id" : "LIBRARY VANDALISM", "Number_crime" : 71 }
```

Use Case 11

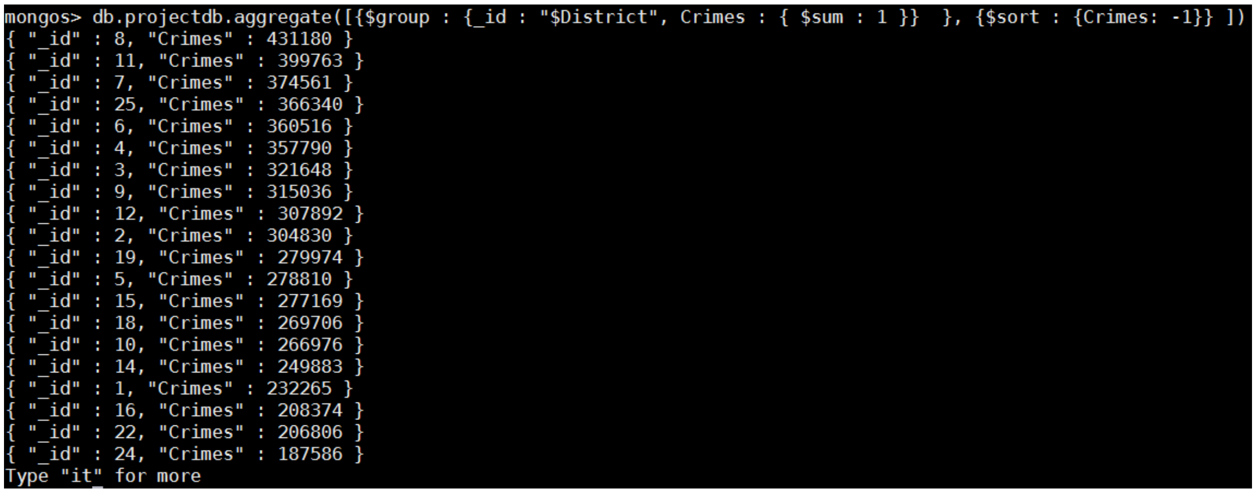
Description: Find case number of all the crimes that happened in the location range given.
Query: db.projectdb.find({ Latitude: { $gte: 41.79922968 }, Longitude: { $lte: -87.766148551 } }, { "Case Number":1 }).pretty()
Output:



```
mongos> db.projectdb.find({ Latitude: { $gte: 41.79922968 }, Longitude:{$lte:-87.766148551} },{"Case Number":1}).pretty()
{ "_id" : ObjectId("590e33b6beca921a93ab6871"), "Case Number" : "HN663119" }
{ "_id" : ObjectId("590e33b6beca921a93ab68c0"), "Case Number" : "HN663224" }
{ "_id" : ObjectId("590e33b6beca921a93ab68c1"), "Case Number" : "HN665136" }
{ "_id" : ObjectId("590e33b6beca921a93ab695a"), "Case Number" : "HN664042" }
{ "_id" : ObjectId("590e33b6beca921a93ab697e"), "Case Number" : "HN663947" }
{ "_id" : ObjectId("590e33b6beca921a93ab6986"), "Case Number" : "HN665716" }
{ "_id" : ObjectId("590e33b6beca921a93ab698c"), "Case Number" : "HN662164" }
{ "_id" : ObjectId("590e33b6beca921a93ab6998"), "Case Number" : "HN661042" }
{ "_id" : ObjectId("590e33b6beca921a93ab699a"), "Case Number" : "HN661594" }
{ "_id" : ObjectId("590e33b6beca921a93ab69ae"), "Case Number" : "HN665941" }
{ "_id" : ObjectId("590e33b6beca921a93ab69b4"), "Case Number" : "HN665968" }
{ "_id" : ObjectId("590e33b6beca921a93ab69cc"), "Case Number" : "HN666034" }
{ "_id" : ObjectId("590e33b6beca921a93ab69da"), "Case Number" : "HN666187" }
{ "_id" : ObjectId("590e33b6beca921a93ab69fb"), "Case Number" : "HN666376" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a01"), "Case Number" : "HN665780" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a05"), "Case Number" : "HN665341" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a0c"), "Case Number" : "HN665420" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a24"), "Case Number" : "HN666074" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a40"), "Case Number" : "HN664870" }
{ "_id" : ObjectId("590e33b6beca921a93ab6a49"), "Case Number" : "HN663191" }
Type "it" for more
```

Use Case 12

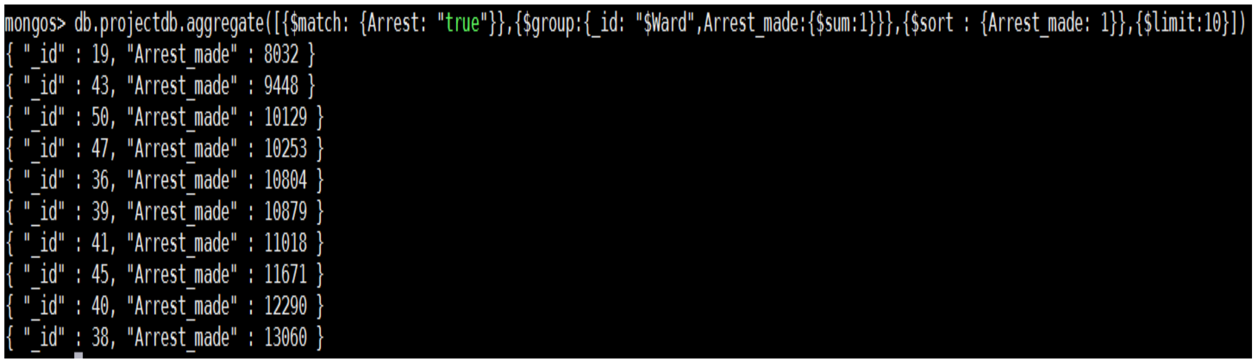
Description: Most unsafe districts in descending order.
Query: db.projectdb.aggregate([{ $group: { _id: "$District", Crimes: { $sum: 1 } } }, { $sort: { Crimes: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$group : {_id : "$District", Crimes : { $sum : 1 }}  }, {$sort : {Crimes: -1}} ])
{ "_id" : 8, "Crimes" : 431180 }
{ "_id" : 11, "Crimes" : 399763 }
{ "_id" : 7, "Crimes" : 374561 }
{ "_id" : 25, "Crimes" : 366340 }
{ "_id" : 6, "Crimes" : 360516 }
{ "_id" : 4, "Crimes" : 357790 }
{ "_id" : 3, "Crimes" : 321648 }
{ "_id" : 9, "Crimes" : 315036 }
{ "_id" : 12, "Crimes" : 307892 }
{ "_id" : 2, "Crimes" : 304830 }
{ "_id" : 19, "Crimes" : 279974 }
{ "_id" : 5, "Crimes" : 278810 }
{ "_id" : 15, "Crimes" : 277169 }
{ "_id" : 18, "Crimes" : 269706 }
{ "_id" : 10, "Crimes" : 266976 }
{ "_id" : 14, "Crimes" : 249883 }
{ "_id" : 1, "Crimes" : 232265 }
{ "_id" : 16, "Crimes" : 208374 }
{ "_id" : 22, "Crimes" : 206806 }
{ "_id" : 24, "Crimes" : 187586 }
Type "it" for more
```

Use Case 13

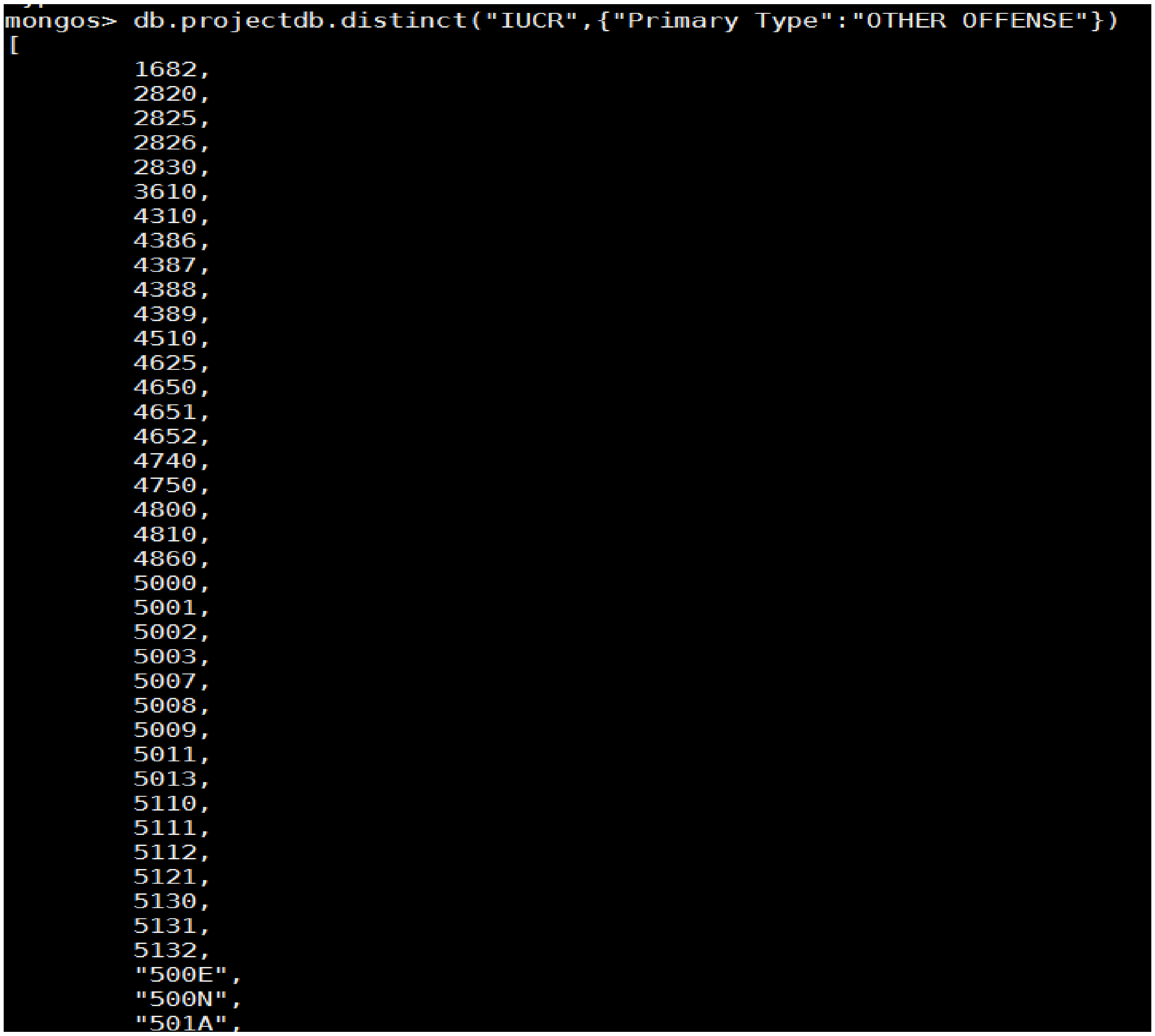
Description: first 10 wards of police that made least number of arrest.
Query: db.projectdb.aggregate([{ $match: { Arrest: "true" } }, { $group: { _id: "$Ward", Arrest_made: { $sum: 1 } } }, { $sort: { Arrest_made: 1 } }, { $limit:10 }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {Arrest: "true"}},{$group:{_id: "$Ward",Arrest_made:{$sum:1}}},{$sort : {Arrest_made: 1}},{$limit:10}])
{ "_id" : 19, "Arrest_made" : 8032 }
{ "_id" : 43, "Arrest_made" : 9448 }
{ "_id" : 50, "Arrest_made" : 10129 }
{ "_id" : 47, "Arrest_made" : 10253 }
{ "_id" : 36, "Arrest_made" : 10804 }
{ "_id" : 39, "Arrest_made" : 10879 }
{ "_id" : 41, "Arrest_made" : 11018 }
{ "_id" : 45, "Arrest_made" : 11671 }
{ "_id" : 40, "Arrest_made" : 12290 }
{ "_id" : 38, "Arrest_made" : 13060 }
```

Use Case 14

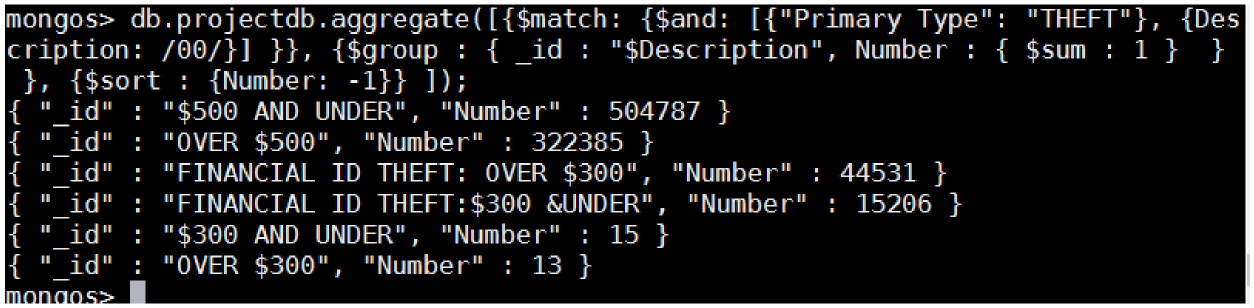
Description: Find distinct IUCR code of all the crime under ‘OTHER OFFENSE’ category
Query: db.projectdb.distinct("IUCR", { "Primary Type": "OTHER OFFENSE" })
Output:



```
mongos> db.projectdb.distinct("IUCR",{"Primary Type":"OTHER OFFENSE"})
[
        1682,
        2820,
        2825,
        2826,
        2830,
        3610,
        4310,
        4386,
        4387,
        4388,
        4389,
        4510,
        4625,
        4650,
        4651,
        4652,
        4740,
        4750,
        4800,
        4810,
        4860,
        5000,
        5001,
        5002,
        5003,
        5007,
        5008,
        5009,
        5011,
        5013,
        5110,
        5111,
        5112,
        5121,
        5130,
        5131,
        5132,
        "500E",
        "500N",
        "501A",
```

Use Case 15

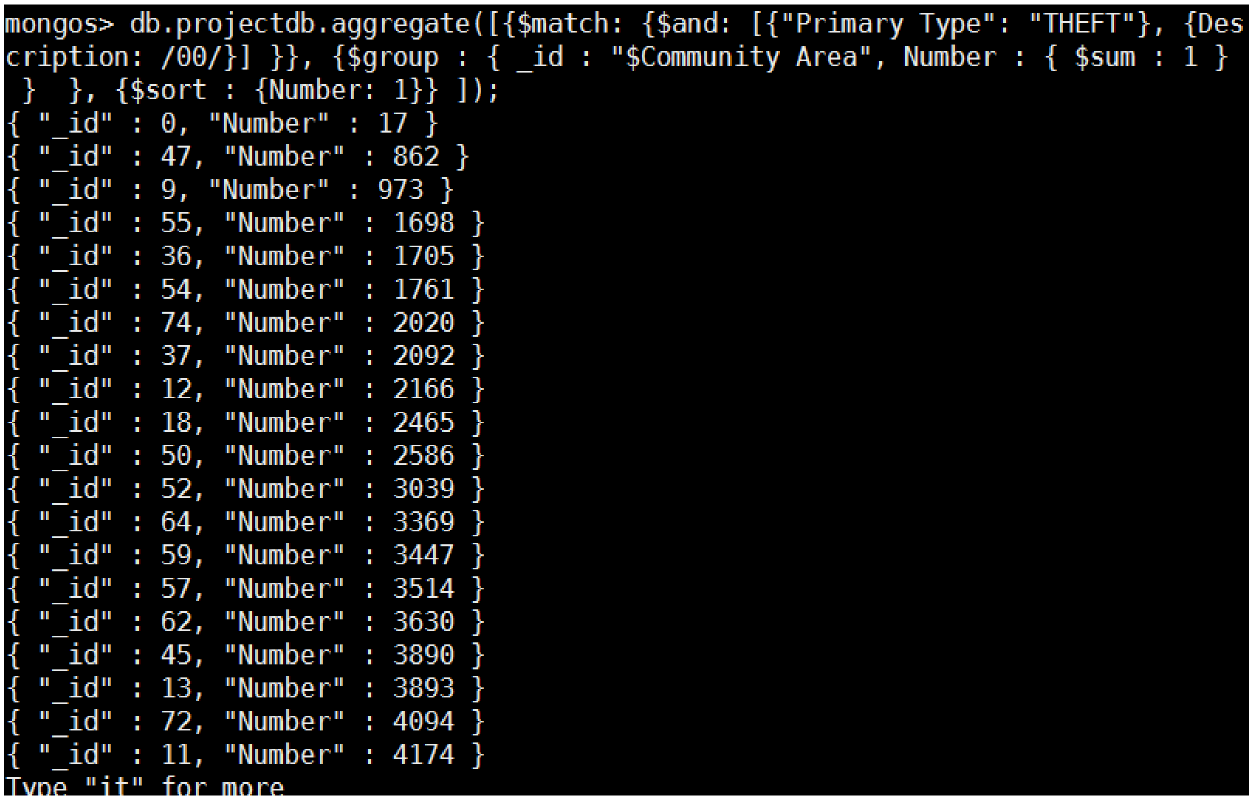
Description: Type and number of Theft crime involving money specified in amount.
Query: db.projectdb.aggregate([{ $match: { $and: [{ "Primary Type": "THEFT" }, { Description: /00/ }] } }, { $group: { _id : "$Description", Number: { $sum: 1 } } }, { $sort: { Number: -1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {$and: [{"Primary Type": "THEFT"}, {Des
cription: /00/}] }}, {$group : { _id : "$Description", Number : { $sum : 1 }  }
 }, {$sort : {Number: -1}} ]);
{ "_id" : "$500 AND UNDER", "Number" : 504787 }
{ "_id" : "OVER $500", "Number" : 322385 }
{ "_id" : "FINANCIAL ID THEFT: OVER $300", "Number" : 44531 }
{ "_id" : "FINANCIAL ID THEFT:$300 &UNDER", "Number" : 15206 }
{ "_id" : "$300 AND UNDER", "Number" : 15 }
{ "_id" : "OVER $300", "Number" : 13 }
mongos>
```

Use Case 16

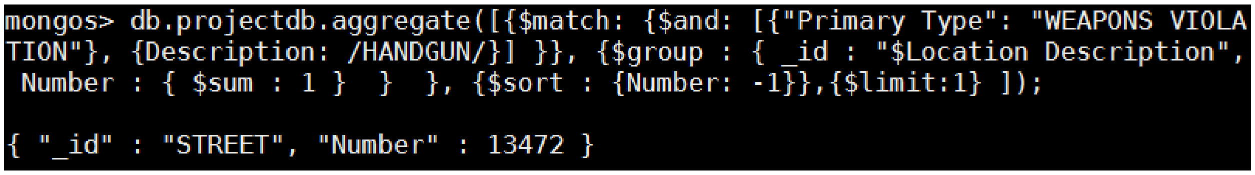
Description: Community areas with with least number of Theft Crime involving amount of money in ascending order.
Query: db.projectdb.aggregate([{ $match: { $and: [{ "Primary Type": "THEFT" }, { Description: /00/ }] } }, { $group: { _id: "$Community Area", Number: { $sum: 1 } } }, { $sort: { Number: 1 } }])
Output:



```
mongos> db.projectdb.aggregate([{$match: {$and: [{"Primary Type": "THEFT"}, {Des
cription: /00/}] }}, {$group : { _id : "$Community Area", Number : { $sum : 1 }
 }  }, {$sort : {Number: 1}} ]);
{ "_id" : 0, "Number" : 17 }
{ "_id" : 47, "Number" : 862 }
{ "_id" : 9, "Number" : 973 }
{ "_id" : 55, "Number" : 1698 }
{ "_id" : 36, "Number" : 1705 }
{ "_id" : 54, "Number" : 1761 }
{ "_id" : 74, "Number" : 2020 }
{ "_id" : 37, "Number" : 2092 }
{ "_id" : 12, "Number" : 2166 }
{ "_id" : 18, "Number" : 2465 }
{ "_id" : 50, "Number" : 2586 }
{ "_id" : 52, "Number" : 3039 }
{ "_id" : 64, "Number" : 3369 }
{ "_id" : 59, "Number" : 3447 }
{ "_id" : 57, "Number" : 3514 }
{ "_id" : 62, "Number" : 3630 }
{ "_id" : 45, "Number" : 3890 }
{ "_id" : 13, "Number" : 3893 }
{ "_id" : 72, "Number" : 4094 }
{ "_id" : 11, "Number" : 4174 }
Type "it" for more
```

Use Case 17

Description: In which location did most attacks with Handgun happened.
Query: db.projectdb.aggregate([{ $match: { $and: [{ "Primary Type": "WEAPONS VIOLATION" }, { Description: /HANDGUN/ }] } }, { $group: { _id: "$Location Description", Number: { $sum: 1 } } }, { $sort: { Number: -1 } }, { $limit: 1 }])
Output:

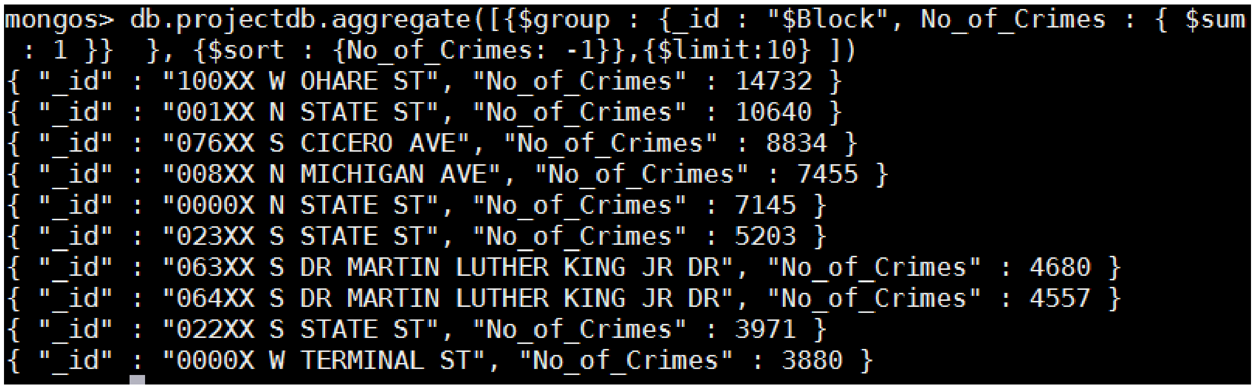


```
mongos> db.projectdb.aggregate([{$match: {$and: [{"Primary Type": "WEAPONS VIOLA
TION"}, {Description: /HANDGUN/}] }}, {$group : { _id : "$Location Description",
 Number : { $sum : 1 }  }  }, {$sort : {Number: -1}},{$limit:1} ]);

{ "_id" : "STREET", "Number" : 13472 }
```

Use Case 18

Description : top 10 most unsafe blocks in descending order.

Query: db.projectdb.aggregate([{ $group: { _id: "$Block", No_of_Crimes: { $sum: 1 } } }, { $sort: { No_of_Crimes: -1 } }, { $limit:10 }])
Output:



```
mongos> db.projectdb.aggregate([{$group : {_id : "$Block", No_of_Crimes : { $sum
 : 1 }}  }, {$sort : {No_of_Crimes: -1}},{$limit:10} ])
{ "_id" : "100XX W OHARE ST", "No_of_Crimes" : 14732 }
{ "_id" : "001XX N STATE ST", "No_of_Crimes" : 10640 }
{ "_id" : "076XX S CICERO AVE", "No_of_Crimes" : 8834 }
{ "_id" : "008XX N MICHIGAN AVE", "No_of_Crimes" : 7455 }
{ "_id" : "0000X N STATE ST", "No_of_Crimes" : 7145 }
{ "_id" : "023XX S STATE ST", "No_of_Crimes" : 5203 }
{ "_id" : "063XX S DR MARTIN LUTHER KING JR DR", "No_of_Crimes" : 4680 }
{ "_id" : "064XX S DR MARTIN LUTHER KING JR DR", "No_of_Crimes" : 4557 }
{ "_id" : "022XX S STATE ST", "No_of_Crimes" : 3971 }
{ "_id" : "0000X W TERMINAL ST", "No_of_Crimes" : 3880 }
```