

Assignment – 7(802)

MongoDB

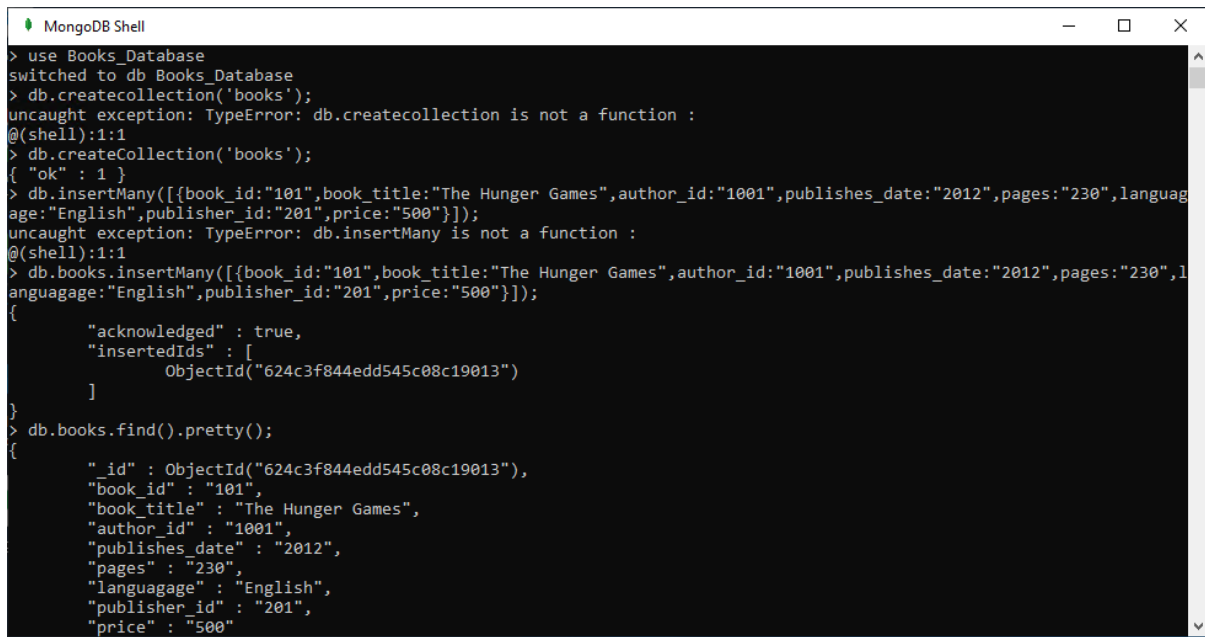
Name :B Karthik

MID:M1082972

Source Code:

<https://github.com/BKarthik1/BKarthik1.gits>

Q1. Write a query to display all the documents presents in **books** collection:



```
MongoDB Shell
> use Books_Database
switched to db Books_Database
> db.createcollection('books');
uncaught exception: TypeError: db.createcollection is not a function :
@(shell):1:1
> db.createCollection('books');
{ "ok" : 1 }
> db.insertMany([{"book_id":"101","book_title":"The Hunger Games",author_id:"1001",publishes_date:"2012",pages:"230",language:"English",publisher_id:"201",price:"500"}]);
uncaught exception: TypeError: db.insertMany is not a function :
@(shell):1:1
> db.books.insertMany([{"book_id":"101","book_title":"The Hunger Games",author_id:"1001",publishes_date:"2012",pages:"230",language:"English",publisher_id:"201",price:"500"}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624c3f844edd545c08c19013")
  ]
}
> db.books.find().pretty();
{
  "_id" : ObjectId("624c3f844edd545c08c19013"),
  "book_id" : "101",
  "book_title" : "The Hunger Games",
  "author_id" : "1001",
  "publishes_date" : "2012",
  "pages" : "230",
  "language" : "English",
  "publisher_id" : "201",
  "price" : "500"
}
```

Q2. Write a query to display all the documents presents in **author** collection:

```
MongoDB Shell
{
  "book_title" : "The Hunger Games",
  "author_id" : "1001",
  "publishes_date" : "2012",
  "pages" : "230",
  "language" : "English",
  "publisher_id" : "201",
  "price" : "500"
}
> db.author.find().pretty();
{
  "_id" : ObjectId("624c416d4edd545c08c19014"), "author_id" : "1001" }
{
  "_id" : ObjectId("624c42184edd545c08c19015"),
  "firstname" : "Suzanne",
  "lastname" : "Collins"
}
{
  "_id" : ObjectId("624c43394edd545c08c19016"),
  "age" : "30",
  "gender" : "female",
  "email_id" : "suzannecollinsgmail.com",
  "phone_no" : "123456789"
}
{
  "_id" : ObjectId("624c43c54edd545c08c19017"),
  "street" : "abc",
  "city" : "cdfgh",
  "state" : "ijklmn",
  "country" : "American"
}
>
```

Q3. Write a query to display all the documents presents in **author** collection:

```
MongoDB Shell
WriteResult({ "nRemoved" : 1 })
> db.author.remove({pub_name:"Scholastic Corporation"});
uncaught exception: SyntaxError: missing ] in index expression :
@(shell):1:52
> db.author.remove({pub_name:"Scholastic Corporation"});
WriteResult({ "nRemoved" : 0 })
> db.author.remove({country:"US"});
WriteResult({ "nRemoved" : 0 })
> db.author.insertMany([{"street":"abc",city:"cdfgh",state:"ijklmn",country:"American"}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624c478d4edd545c08c19019")
  ]
}
> db.publisher.insertMany([{"publisher_id":"201",pub_name:"Scholastic Corporation",country:"US"}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624c47eb4edd545c08c1901a")
  ]
}
> db.publisher.find().pretty();
{
  "_id" : ObjectId("624c47eb4edd545c08c1901a"),
  "publisher_id" : "201",
  "pub_name" : "Scholastic Corporation",
  "country" : "US"
}
>
```

Q4. Write a query to display all of the author whose firstname start with S:

```
MongoDB Shell
{
  "_id" : ObjectId("624c42184edd545c08c19015"),
  "firstname" : "Suzanne",
  "lastname" : "Collins"
}
{
  "_id" : ObjectId("624c43394edd545c08c19016"),
  "age" : "30",
  "gender" : "female",
  "email_id" : "suzannecollinsgmail.com",
  "phone_no" : "123456789"
}
{
  "_id" : ObjectId("624c43c54edd545c08c19017"),
  "street" : "abc",
  "city" : "cdfgh",
  "state" : "ijklmn",
  "country" : "American"
}
{
  "_id" : ObjectId("624c478d4edd545c08c19019"),
  "street" : "abc",
  "city" : "cdfgh",
  "state" : "ijklmn",
  "country" : "American"
}
> db.col.find({firstname:{$regex:'S'}});
> db.author.find({firstname:{$regex:'S'}});
{ "_id" : ObjectId("624c42184edd545c08c19015"), "firstname" : "Suzanne", "lastname" : "Collins" }
>
```

Q5. Write a query to display all the details of the book published date:

```
Country : American
}
> db.col.find({firstname:{$regex:'S'}});
> db.author.find({firstname:{$regex:'S'}});
{ "_id" : ObjectId("624c42184edd545c08c19015"), "firstname" : "Suzanne", "lastname" : "Collins" }
> db.author.find({publishes_date:{$gte:'2012'}});
> db.books.find({publishes_date:{$gte:'2012'}});
{ "_id" : ObjectId("624c3f844edd545c08c19013"), "book_id" : "101", "book_title" : "The Hunger Games", "author_id" : "1001", "publishes_date" : "2012", "pages" : "230", "language" : "English", "publisher_id" : "201", "price" : "500" }
>
```

Q6. Write a query to display all the details of the book pages greater than 2 and English as a language:

```
> db.books.find({pages:{$gt:'200'}});
{ "_id" : ObjectId("624c3f844edd545c08c19013"), "book_id" : "101", "book_title" : "The Hunger Games", "author_id" : "1001", "publishes_date" : "2012", "pages" : "230", "language" : "English", "publisher_id" : "201", "price" : "500" }
>
```

Q7. Write a query to display all the details of the authors age greater than 30 and less than 60:

```
> db.author.find({age:{$gt:'20'}}).pretty();
{
  "_id" : ObjectId("624c43394edd545c08c19016"),
  "age" : "30",
  "gender" : "female",
  "email_id" : "suzanncollinsgmail.com",
  "phone_no" : "123456789"
}
```

Q8. . Write a query to display all of the author whose last name start with Collins:

```
> db.author.find({lastname:{$regex:'Collins'}}).pretty();
{
  "_id" : ObjectId("624c42184edd545c08c19015"),
  "firstname" : "Suzanne",
  "lastname" : "Collins"
}
```

Q9. Write a query to display the title of all the documents present in the book collection.

```
> db.books.find({}, {title:"The hunger games"}).pretty();
{
  "_id" : ObjectId("624c3f844edd545c08c19013"),
  "title" : "The hunger games"
}
```

Q10. Write a query to display author of all the documents present in the last name :

```
> db.books.find({}, {lastname:"The hunger games"}).pretty();
{
  "_id" : ObjectId("624c3f844edd545c08c19013"),
  "lastname" : "The hunger games"
}
```

Q11. Create a database with the above automobile structure in Mongo Shell:

```

> db.automobile.find();
{
  "_id" : "NEXON", "ancestors" : [ "AUTOMOBILE", "CAR", "TATA" ], "parent" : "TATA" }
{
  "_id" : "SWIFT", "ancestors" : [ "AUTOMOBILE", "CAR", "MARUTI" ], "parent" : "MARUTI" }
{
  "_id" : "TATA", "ancestors" : [ "AUTOMOBILE", "CAR" ], "parent" : "CAR" }
{
  "_id" : "MARUTI", "ancestors" : [ "AUTOMOBILE", "CAR" ], "parent" : "CAR" }
{
  "_id" : "CAR", "ancestors" : [ "AUTOMOBILE" ], "parent" : "AUTOMOBILE" }
{
  "_id" : "AUTOMOBILE", "ancestors" : [ ], "parent" : null }
{
  "_id" : "SCOOTY", "ancestors" : [ "AUTOMOBILE", "BIKE", "TVS" ], "parent" : "TVS" }
{
  "_id" : "APACHE", "ancestors" : [ "AUTOMOBILE", "BIKE", "TVS" ], "parent" : "TVS" }
{
  "_id" : "SHINE", "ancestors" : [ "AUTOMOBILE", "BIKE", "HONDA" ], "parent" : "HONDA" }
{
  "_id" : "UNICORN", "ancestors" : [ "AUTOMOBILE", "BIKE", "HONDA" ], "parent" : "HONDA" }
{
  "_id" : "TVS", "ancestors" : [ "AUTOMOBILE", "BIKE" ], "parent" : "BIKE" }
{
  "_id" : "HONDA", "ancestors" : [ "AUTOMOBILE", "BIKE" ], "parent" : "BIKE" }
{
  "_id" : "BIKE", "ancestors" : [ "AUTOMOBILE" ], "parent" : "AUTOMOBILE" }

```

Q12. Insert document with Vehicle_no, Owner_name, O_city with relevant values for the HONDA collection:

```

> db.HONDA.insertMany([{"vehicle_no":"TN 04 p xxxx"}, {"owner_name":"Steve"}, {"O_city":"Tokyo"}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("624c8cff3e431304c26d6f91"),
    ObjectId("624c8cff3e431304c26d6f92"),
    ObjectId("624c8cff3e431304c26d6f93")
  ]
}

```

Q13. Write a query to retrieve the ancestors of the nodes:

```

> db.automobile.findOne({ancestors:"AUTOMOBILE"});
{
  "_id" : "NEXON",
  "ancestors" : [
    "AUTOMOBILE",
    "CAR",
    "TATA"
  ],
  "parent" : "TATA"
}
>

```

Q14, Write a query to add a new node as BREEZA under Maruti:

```

> db.automobile.insert({_id:"BREEZA", ancestors:["AUTOMOBILE","CAR","MARUTI"],parent:"MARUTI"});
WriteResult({ "nInserted" : 1 })
> db.automobile.find({});
{ "_id" : "NEXON", "ancestors" : [ "AUTOMOBILE", "CAR", "TATA" ], "parent" : "TATA" }
{ "_id" : "SWIFT", "ancestors" : [ "AUTOMOBILE", "CAR", "MARUTI" ], "parent" : "MARUTI" }
{ "_id" : "TATA", "ancestors" : [ "AUTOMOBILE", "CAR" ], "parent" : "CAR" }
{ "_id" : "MARUTI", "ancestors" : [ "AUTOMOBILE", "CAR" ], "parent" : "CAR" }
{ "_id" : "CAR", "ancestors" : [ "AUTOMOBILE" ], "parent" : "AUTOMOBILE" }
{ "_id" : "AUTOMOBILE", "ancestors" : [ ], "parent" : null }
{ "_id" : "SCOOTY", "ancestors" : [ "AUTOMOBILE", "BIKE", "TVS" ], "parent" : "TVS" }
{ "_id" : "APACHE", "ancestors" : [ "AUTOMOBILE", "BIKE", "TVS" ], "parent" : "TVS" }
{ "_id" : "SHINE", "ancestors" : [ "AUTOMOBILE", "BIKE", "HONDA" ], "parent" : "HONDA" }
{ "_id" : "UNICORN", "ancestors" : [ "AUTOMOBILE", "BIKE", "HONDA" ], "parent" : "HONDA" }
{ "_id" : "TVS", "ancestors" : [ "AUTOMOBILE", "BIKE" ], "parent" : "BIKE" }
{ "_id" : "HONDA", "ancestors" : [ "AUTOMOBILE", "BIKE" ], "parent" : "BIKE" }
{ "_id" : "BIKE", "ancestors" : [ "AUTOMOBILE" ], "parent" : "AUTOMOBILE" }
{ "_id" : "BREEZA", "ancestors" : [ "AUTOMOBILE", "CAR", "MARUTI" ], "parent" : "MARUTI" }
>

```

Q15. Write a query to get children nodes of CARS:

```

> db.automobile.insert({_id:"CAR", children:["MARUTI","TATA"]});
WriteResult({ "nInserted" : 1 })
> db.automobile.findOne({_id:"CAR"}).children;
[ "MARUTI", "TATA" ]
>

```

Q16. Write a pros and cons for RDBMS and NoSql Database:



RDBMS Vs. NoSQL

RDBMS

- Structured and organized data
- Structured Query Language (SQL)
- Data and its relationships stored in separate tables.
- Data Manipulation Language, Data Definition Language
- Tight Consistency
- BASE Transaction

NoSQL

- No declarative query language
- No predefined schema
- Key-Value pair storage, Column Store, Document Store, Graph Databases
- Eventual consistency rather ACID property
- Unstructured and unpredictable data
- CAP Theorem
- Prioritize high performance, high availability and scalability

Q17.

NoSQL:

```
> db.SPORT.insert({PLAYER_ID:"5001"},{PLAYER_NAME:"MSD"},{PLAYER_COUNTRY:"IND"},{PLAYER_SPORT_ID:"1"},{PLAYER_SPORT_NAME:"CRICKET"},{PLAYER_SPORT_COUNTRY:"IND"});
WriteResult({ "nInserted" : 1 })
> db.SPORT.insert({PLAYER_ID:"5002"},{PLAYER_NAME:"VIRAT"},{PLAYER_COUNTRY:"IND"},{PLAYER_SPORT_ID:"1"},{PLAYER_SPORT_NAME:"CRICKET"},{PLAYER_SPORT_COUNTRY:"IND"});
WriteResult({ "nInserted" : 1 })
> db.SPORT.insert({PLAYER_ID:"9001"},{PLAYER_NAME:"LIONEL MESSI"},{PLAYER_COUNTRY:"ARG"},{PLAYER_SPORT_ID:"2"},{PLAYER_SPORT_NAME:"FOOTBALL"},{PLAYER_SPORT_COUNTRY:"ARG"});
WriteResult({ "nInserted" : 1 })
> db.SPORT.insert({PLAYER_ID:"9002"},{PLAYER_NAME:"CRISTIANO RONALDO"},{PLAYER_COUNTRY:"PORT"},{PLAYER_SPORT_ID:"2"},{PLAYER_SPORT_NAME:"FOOTBALL"},{PLAYER_SPORT_COUNTRY:"PORT"});
WriteResult({ "nInserted" : 1 })
> db.SPORT.insert({PLAYER_ID:"9003"},{PLAYER_NAME:"RAFAEL NADAL"},{PLAYER_COUNTRY:"SPAIN"},{PLAYER_SPORT_ID:"2"},{PLAYER_SPORT_NAME:"FOOTBALL"},{PLAYER_SPORT_COUNTRY:"SPAIN"});
WriteResult({ "nInserted" : 1 })
> db.SPORT.find();
{ "_id" : ObjectId("624d00e503863f53947a4da7"), "PLAYER_ID" : "5001" }
{ "_id" : ObjectId("624d00fc03863f53947a4da8"), "PLAYER_ID" : "5002" }
{ "_id" : ObjectId("624d012f03863f53947a4da9"), "PLAYER_ID" : "9001" }
{ "_id" : ObjectId("624d015603863f53947a4daa"), "PLAYER_ID" : "9002" }
{ "_id" : ObjectId("624d017d03863f53947a4dab"), "PLAYER_ID" : "9003" }
```

SQL:

use player

create table SPORTS(

SPORTS_ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

SPORT_NAME NCHAR(50));

DROP TABLE SPORTS

CREATE TABLE PLAYERS(

PLAYER_ID INT,

PLAYER_NAME VARCHAR(50),

PLAYER_COUNTRY VARCHAR(20),

PLAYER_SPORT_ID INT);

INSERT INTO SPORTS(SPORT_NAME)

VALUES('CRICKET'),('FOOTBALL'),('TENNIS');

	SPORTS_ID	SPORT_NAME
1	1	CRICKET
2	2	FOOTBALL
3	3	TENNIS

INSERT INTO

PLAYERS(PLAYER_ID,PLAYER_NAME,PLAYER_COUNTRY,PLAYER_SPORT_ID)

```
VALUES(5001,'MSD','IND',1),
(5002,'VIRAT','IND',1),
(9001,'LIONEL MESSI','ARG',2),
(9002,'CRISTIANO RONALDO','PORT',2),
(9003,'RAFAEL NADAL','SPAIN',3);
```

	PLAYER_ID	PLAYER_NAME	PLAYER_COUNTRY	PLAYER_SPORT_ID
1	5001	MSD	IND	1
2	5002	VIRAT	IND	1
3	9001	LIONEL MESSI	ARG	2
4	9002	CRISTIANO RONALDO	PORT	2
5	9003	RAFAEL NADAL	SPAIN	3

Q18.

NOSQL:

```
{ "_id" : ObjectId( "6240017d05803f53947a4da0" ), "PLAYER_ID" : "9003" }
> db.PLAYER.updateOne({PLAYER_ID:"5002"},{$rename:{"VIRAT":"ROHIT"}});
{ "acknowledged" : true, "matchedCount" : 0, "modifiedCount" : 0 }
> db.PLAYER.find();
```

SQL:

```
ALTER TABLE PLAYERS RENAME COLUMN 'VIRAT' TO 'ROHIT' ;
```

19.

NOSQL:

```
> db.SPORT.count({});
5
> _
```

SQL:

```
SELECT COUNT(PLAYER_ID) FROM PLAYERS
```


	(No column name)	
1	5	

20.

NOSQL:

```
> db.SPORT.find({});
{ "_id" : ObjectId("624d00e503863f53947a4da7"), "PLAYER_ID" : "5001" }
{ "_id" : ObjectId("624d00fc03863f53947a4da8"), "PLAYER_ID" : "5002" }
```

SQL:

```
SELECT COUNT(PLAYER_ID),PLAYER_COUNTRY FROM PLAYERS GROUP BY PLAYER_COUNTRY
HAVING COUNT(PLAYER_COUNTRY)>1
```

	(No column name)	PLAYER_COUNTRY
1	2	IND