Q1. Which of the following query selects all documents in the records collection that match the condition { "user\_id": { $lt: 42 } } ?

1. db.records.findOne( { "user\_id": { $lt: 42 } } )
2. **db.records.find( { "user\_id": { $lt: 42 } })**
3. db.records.findOne( { "user\_id": { $lt: 42 } })
4. db.records.select( { "user\_id": { $lt: 42 } })

Q2. Which of the following is not a projection operator?

1. $slice
2. $elemMatch
3. $split
4. **None of the mentioned**

Q3. $multiply \_\_\_\_\_\_\_\_\_\_ numbers to return the result.

1. **multiplies**
2. divides
3. comapres
4. All of the mentioned

Q4. \_\_\_\_\_\_\_ adds numbers to return the sum, or adds numbers and a date to return a new date.

1. $addnum
2. $setadd
3. $adds
4. **$add**

Q5. \_\_\_\_\_\_\_ returns the remainder of the first number divided by the second.

1. $rem
2. **$mod**
3. $div
4. None of the mentioned

Q6. What does the following command do?

db.demo.find().limit(10)

1. Show ten documents randomly from the collection demo
2. **Show only first ten documents from the collection demo**
3. Repeats the first document ten times
4. All of the above.

Q7. Which of the following command is correct when you want to fetch documents form collection demo, where value of a field 'interest' is null?

1. **db.demo.find( { "interest" : null } )**
2. db.demo.find( { "interest : null" } )
3. db.demo.find().sort( { "interest" : null } )
4. Either A or B

Q8. Which of the following is correct command to insert data into MongoDB?

Assume that document is a valid JSON document.

1. employees.insert(document)
2. db.employees.insert().(document)
3. **db.employees.insert(document)**
4. db.insert.employee(document)

Q9. Which of the following command is correct when you want to fetch documents from collection only employees whose salary is either 7500 or 10,000?

1. db.employees.find({"salary" :{"$in :[7500, 10000]"}})
2. db.employees.find.sort({"salary" :{$in :[7500, 10000]}})
3. **db.employees.find({"salary" :{$in :[7500, 10000]}})**
4. db.find.employees({"salary" :{$in :[7500, 10000]}})

Q10. Which one of the following is equivalent to?

Select \* from employees order by salary

1. db.employees.sort({"salary":1})
2. db.employees.find().sort({"salary:1"})
3. **db.employees.find().sort({"salary":1})**
4. db.employees.find().order({"salary":1})

Q11. Which of the following answers equals to SQL command – SELECT emp\_id from EMPLOYEES where designation="Manager";

1. employees.find({"designation":"manager"}
2. **db.employees.find({"designation":"manager"}**
3. db.employees.find({"designation:manager"}
4. Any one of the above.

Q12. $set’ is used for \_\_\_\_\_\_\_\_\_\_\_.

1. Delete
2. **Update**
3. Insert
4. All of the above.

Q13. What does the following command do?

db.employees.find().skip(5).limit(5)

1. **Skips first five documents and then shows just next five documents**
2. Shows just next five documents
3. Skips first five documents and then shows the sixth one five times
4. None of the above

Q14. Which one of the following is equivalent to?

SELECT \* from EMPLOYEES order by salary desc;

1. **db.employees.find().sort({"salary":-1})**
2. db.employees.sort({"salary":-1})
3. db.employees.find().sort({"salary":1})
4. db.employees.find().sort({"salary": desc})

Q15. If you have created a compound index on (P, Q, R) which of the following access pattern will not be able to utilize the index?

1. P, Q, R
2. P, Q
3. **Q, R**
4. P

Q16. Which of the following is a valid MongoDB statement?

1. **db.author.insertOne( { usrName : "John Doe", usrDept : "Sales", usrTitle : "Executive Account Manager", authLevel : 4, authDept : [ "Sales", "Customers"] } )**
2. db.author.insertOne( { usrName : "John Doe"; usrDept : "Sales"; usrTitle : "Executive Account Manager"; authLevel : 4; authDept : [ "Sales", "Customers"] } )
3. db.author.insertOne( { usrName = "John Doe", usrDept = "Sales", usrTitle = "Executive Account Manager", authLevel = 4, authDept = [ "Sales", "Customers"] } )
4. None of the above.

Q17. Which of the following is a valid MongoDB statement to insert a single document?

1. **db.author.insertOne( { usrName: "John Doe", usrDept : "Sales"} )**
2. db.author.insertone( { usrName : "John Doe", usrDept : "Sales"} )
3. db.author.insertOne( { usrName = "John Doe", usrDept = "Sales"} )
4. db.author.insert1( { usrName : "John Doe", usrDept : "Sales" } )

Q18.

Q19.

Q20.

Q21.

Q22.

Q23.

Q24.

Q25.

Q26.

Q27.

Q28.

Q29.

Q30.

Q31.

Q32.

Q33.

Q34.

Q35.

Q36.

Q37.

Q38.

Q39.

Q40.

Q41.

Q42.

Q43.

Q44.

Q45.

Q46.

Q47.

Q48.

Q49.

Q50.

Q51.

Q52.

Q53.

Q54.

Q55.

Q56.

Q57.

Q58.

Q59.

Q60.

Q61.

Q62.

Q63.

Q64.

Q65.

Q66.

Q67.

Q68.

Q69.

Q70.

Q71.

Q72.

Q73.

Q74.

Q75.

Q76.

Q77.

Q78.

Q79.

Q80.

Q81.

Q82.

Q83.

Q84.

Q85.

Q86.

Q87.

Q88.

Q89.

Q90.