Q1. Point out the wrong statement:

1. **Indexes cannot enforce uniqueness in collection**
2. CRUD stands for create, read, update, and delete
3. CRUD application is the most simplest application
4. All of the mentioned

Q2. \_\_\_\_\_\_ returns true if the first value is less than the second.

1. $lte
2. **$lt**
3. $setEquals
4. None of the mentioned

Q3. $ne returns true if the values are not \_\_\_\_\_\_\_ .

1. **equivalent**
2. null
3. 0
4. None of the mentioned

Q4. In $min operator, if the key is having null, than what value $min will returns.

1. 1
2. 0
3. **null**
4. None of the mentioned

Q5. In $max operator, if the key is having null, than what value $max will returns.

1. 1
2. 0
3. **null**
4. None of the mentioned

Q6. In $sum operator, if the key is having null, than what value $sum will returns.

1. 1
2. **0**
3. null
4. None of the mentioned

Q7. Which of the following is equivalent to this?

SELECT first\_name, salary FROM employees WHERE designation=”Manager”;

Assume that there are three columns first\_name, salary, date\_of\_join.

1. **db.employees.find({"designation":"Manager"},{"date\_of\_join" : 0})**
2. db.employees.find({"designation:Manager"},{"date\_of\_join" : 0})
3. db.employees.find({"designation:Manager"},{"date\_of\_join" : 1})
4. None of the above.

Q8. Which of the following MongoDB conditional operator is not a valid operator?

1. $lte
2. $gt
3. **$gtr**
4. $lt

Q9. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM emp WHERE deptno = 30 ORDER BY deptno DESC;

1. **db.emp.find( {deptno: 30 } ).sort({ deptno: -1 } )**
2. db.emp.find( {deptno= 30 } ).sort({ deptno: 1 } )
3. db.emp.find( {deptno: 30 } ).orderby({ deptno: -1 } )
4. an error in statement.

Q10. What is equivalent MongoDB query for the following SQL statement.

INSERT INTO emp values("abc123", 55, "A");

1. db.people.Insert( { user\_id: "abc123", age: 55, status: "A" } )
2. db.people.insertone( { user\_id: "abc123", age: 55, status: "A" } )
3. **db.people.insertOne( { user\_id: "abc123", age: 55, status: "A" } )**
4. an error in statement.

Q11. What is equivalent MongoDB query for the following SQL statement.

ALTER TABLE people DROP COLUMN join\_date;

1. **db.people.updateMany({ }, { $unset: { "join\_date": "" } })**
2. db.people.updateMany({ }, { $unSet: { "join\_date": "" } })
3. db.people.updateMany({ }, { $unset: { "join\_date": Null } })
4. an error in statement.

Q12. What is equivalent MongoDB query for the following SQL statement.

ALTER TABLE people ADD join\_date DATETIME

1. **db.people.updateMany({ }, { $set: { join\_date: new Date() } })**
2. db.people.updateMany({ }, { $upsert: { join\_date: new Date() } })
3. db.people.updateMany({ }, { $add: { join\_date: new Date() } })
4. an error in statement.

Q13. What is equivalent MongoDB query for the following SQL statement.

DROP TABLE people;

1. **db.people.drop()**
2. db.people.remove()
3. db.drop.people()
4. an error in statement.

Q14. What is equivalent MongoDB query for the following SQL statement.

INSERT INTO people(user\_id, age, status) VALUES ("bcd001", 45, "A");

1. db.people({ user\_id: "bcd001", age: 45, status: "A" }).insertOne()
2. db.insertOne.people({ user\_id: "bcd001", age: 45, status: "A" })
3. **db.people.insertOne({ user\_id: "bcd001", age: 45, status: "A" })**
4. Both B and C

Q15. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM movies;

1. **db.movies.find();**
2. db.findMovies();
3. db.find.Movies();
4. an error in statement.

Q16. What is equivalent MongoDB query for the following SQL statement.

SELECT \_id, user\_id, status FROM people;

1. db.people.find({ },{ user\_id: True, status: true } );
2. **db.people.find({ },{ user\_id: 1, status: 1 } );**
3. Both A and B;
4. an error in statement.

Q17. What is equivalent MongoDB query for the following SQL statement.

SELECT user\_id, status FROM people;

1. db.people.find({ }, {user\_id: 1, status: 1})
2. db.people.find({ }, {user\_id: 1, status: true, \_id: 0})
3. db.people.find({ }, {user\_id: 1, status: 1, \_id: false})
4. **Both B and C.**

Q18. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A";

1. db.people.findOne({status: "A"})
2. db.people.findMany({status: "A"})
3. **db.people.find({status: "A"})**
4. All of the above.

Q19. What is equivalent MongoDB query for the following SQL statement.

INSERT INTO emp values("abc123", 55, "A");

1. **db.people.insert( { user\_id: "abc123", age: 55, status: "A" } )**
2. db.people.insertone( { user\_id: "abc123", age: 55, status: "A" } )
3. db.people.insertOnes( { user\_id: "abc123", age: 55, status: "A" } )
4. an error in statement.

Q20. What is equivalent MongoDB query for the following SQL statement.

ALTER TABLE people DROP COLUMN join\_date;

1. **db.people.updateMany({ }, { $unset: { "join\_date": 0 } })**
2. db.people.updateMany({ }, { $unset: { "join\_date": Nil } })
3. db.people.updateMany({ }, { $remove: "join\_date" })
4. an error in statement.

Q21. What is equivalent MongoDB query for the following SQL statement.

SELECT user\_id, status FROM people WHERE status = "A";

1. db.people.find({ user\_id: 1, status: 1, \_id: 0 }, { status: "A" })
2. **db.people.find({ status: "A" }, { user\_id: 1, status: 1, \_id: 0 })**
3. db.people.find({$where: { status: "A" }}, { user\_id: 1, status: 1, \_id: 0 })
4. None of the above.

Q22. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status != "A";

1. db.people.find({ status: { $!= "A" } });
2. db.people.find({ $ne: {status: "A" }});
3. **db.people.find({ status: { $ne: "A" } });**
4. None of the above.

Q23. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A" AND age = 50;

1. db.people.find({ status: "A" and age: 50 })
2. db.people.find({ status: "A" $and age: 50 })
3. **db.people.find({ status: "A", age: 50 })**
4. an error in SQL statement.

Q24. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A" OR age = 50;

1. db.people.find({ or: [ { status: "A" } , { age: 50 } ] })
2. db.people.find({ [ { status: "A" } $or: { age: 50 } ] })
3. db.people.find({ [ { status: "A" } , { age: 50 } ] })
4. **None of the above.**

Q25. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age > 25

1. db.people.find({ age: { $gte: 25 } })
2. db.people.find({ age: { $gtr: 25 } })
3. db.people.find({ age: > 25 })
4. **None of the above.**

Q26. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age > 25

1. **db.people.find({ age: { $gt: 25 } })**
2. db.people.find({ age: { $gtr: 25 } })
3. db.people.find({ age: > 25 })
4. None of the above.

Q27. What is equivalent MongoDB query for the following SQL statement.

INSERT INTO emp values("abc123", 55, "A");

1. db.people.insertmany( { user\_id: "abc123", age: 55, status: "A" } )
2. db.people.insertMany( { user\_id: "abc123", age: 55, status: "A" } )
3. **db.people.insertMany( [ { user\_id: "abc123", age: 55, status: "A" } ] )**
4. an error in statement.

Q28. What is equivalent MongoDB query for the following SQL statement.

DROP TABLE people;

1. db.people.Drop()
2. db.people.remove()
3. db.drop.people()
4. **None of the above.**

Q29. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A" OR age = 50;

1. **db.people.find({ $or: [ { status: "A" } , { age: 50 } ] })**
2. db.people.find({ [ { status: "A" } $or: { age: 50 } ] })
3. db.people.find({ [ { status: "A" } , { age: 50 } ] })
4. None of the above.

Q30. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age < 25

1. db.people.find({ age: < 25 })
2. **db.people.find({ age: { $lt: 25 } })**
3. db.people.find({ age: { $less: 25 } })
4. an error in statement.

Q31. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age > 25 AND age <= 50

1. db.people.find({ age: { < 25, < 50 } })
2. db.people.find({ age: { $gt: 25 and $lte: 50 } })
3. db.people.find({ age: { $gt: 25 $and $lte: 50 } })
4. **None of the above.**

Q32. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age > 25 AND age <= 50

1. **db.people.find({ age: { $gt: 25, $lte: 50 } })**
2. db.people.find({ age: { $gt: 25 and $lte: 50 } })
3. db.people.find({ age: { $gt: 25 $and $lte: 50 } })
4. All of the above.

Q33. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE user\_id like "%bc%"

1. **db.people.find( { user\_id: /bc/ } )**
2. db.people.find( { user\_id: { $like:{ 'bc' } } } )
3. db.people.find( { user\_id: $like: 'bc' } )
4. None of the above.

Q34. What is equivalent MongoDB query for the following SQL statement.

SELECT count(\*) FROM people;

1. db.count.people();
2. db.count({people});
3. **db.people.find().count();**
4. None if the above.

Q35. What is equivalent MongoDB query for the following SQL statement.

SELECT count(\*) FROM people;

1. **db.people.count();**
2. db.count({people });
3. db.people.find(count());
4. None if the above.

Q36. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE age < 25

1. db.people.find({ age: < 25 })
2. db.people.find({ age: { $less: 25 } })
3. db.people.find({ age: { less: 25 } })
4. **None of the above.**

Q37. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE user\_id like "%bc%"

1. db.people.find( { user\_id: bc } )
2. db.people.find( { user\_id: { $like:{ 'bc' } } } )
3. db.people.find( { user\_id: $like: 'bc' } )
4. None of the above.

Q38. What is equivalent MongoDB query for the following SQL statement.

SELECT count(\*) FROM people where job='manager';

1. **db.emp.count({job: 'manager'});**
2. db.emp.count({job = 'manager'});
3. db.emp.count({job: $eq: 'manager'});
4. None of the above.

Q39. What is equivalent MongoDB query for the following SQL statement.

SELECT count(\*) FROM people where job='manager';

1. **db.emp.find({job:'manager'}).count();**
2. db.emp.find({job = 'manager'}).count();
3. db.emp.find({job: $eq: 'manager'}).count();
4. None of the above.

Q40. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A" ORDER BY user\_id DESC;

1. db.people.find( { status: "A" } ).sort( { user\_id: $DESC } )
2. db.people.find( { status: "A" } ).sort( { user\_id: $descending } )
3. **db.people.find( { status: "A" } ).sort( { user\_id: -1 } )**
4. Both A and C

Q41. What is equivalent MongoDB query for the following SQL statement.

SELECT \* FROM people WHERE status = "A" ORDER BY user\_id ASC;

1. db.people.find( { status: "A" } ).sort( { user\_id: $ACS } )
2. db.people.find( { status: "A" } ).sort( { user\_id: $Ascending } )
3. **db.people.find( { status: "A" } ).sort( { user\_id: 1 } )**
4. Both A and C

Q42. What is equivalent MongoDB query for the following SQL statement.

SELECT count(comm) FROM emp WHERE comm is not null;

1. db.emp.find( { comm: { $ne: null } } ).count();
2. db.emp.find( { comm != null } ).count();
3. db.emp.find( { comm: { $not: { $eq: null } } } ).count();
4. **Both A and C**

Q43. What is equivalent MongoDB query for the following SQL statement.

SELECT count(comm) FROM emp WHERE comm is null;

1. db.emp.find( { comm: null } ).count();
2. db.emp.find( { comm: { $eq: null } } ).count();
3. db.emp.find( { comm = null } ).count();
4. **Both A and B**

Q44. What is equivalent SQL statement for the following MongoDB query.

db.inventory.find( { qty: { $eq: 20 } } )

1. **SELECT \* FROM inventory WHERE qty = 20;**
2. SELECT qty FROM inventory WHERE qty = 20;
3. Both A and B
4. invalid MongoDB query.

Q45. db.inventory.find( { "item.name": { $eq: "ab" } } )

The query is equivalent to:

db.inventory.find( { "item.name": "ab" } )

1. **True**
2. False

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.