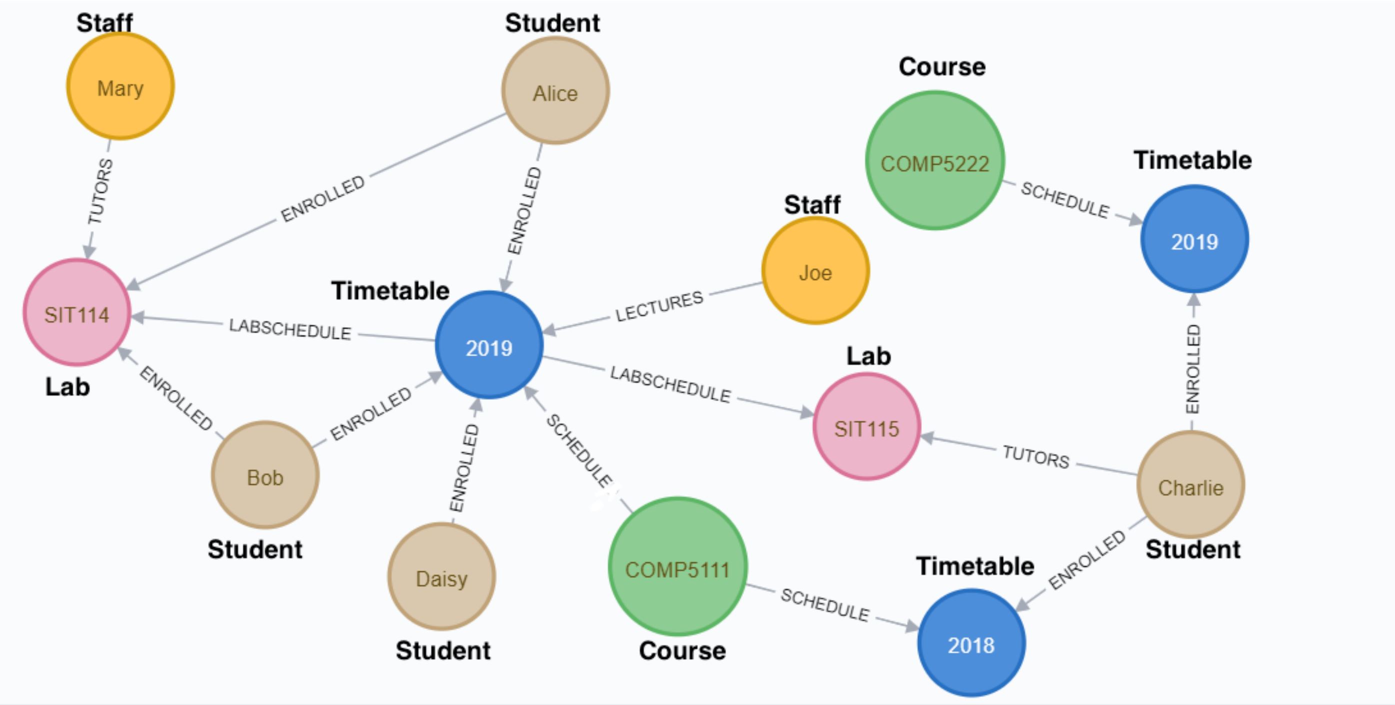
All parts of this question refer to university enrolment data modelled as Neo4j graph. The graph contains five node labels: **Course**, **Timetable**, **Lab**, **Staff** and **Student**. The **Course** node captures basic information about a course. It has two properties: code and title. A course can be offered per year or per semester. Each offering is modelled as a **Timetable** node. The **Timetable** node has four properties: year, semester, venue and time. The **SCHEDULE** relationship between **Course** and **Timetable** captures the course offering information. Most courses have labs, this is modelled as **Lab** node. The **Lab** node has three properties: code and location and time.
The **LABSCHEDULE** relationship is used to indicate which course offering the lab belongs to.
The **Staff** node models academic staff in the system. It has two properties: id and name. Staff can be assigned to course offerings as tutor or lecturer. This is modelled as **LECTURES** or **TUTORS** relationship respectively. The **LECTURES** relationship is between **Staff** and **Timetable** node; while the **TUTORS** relationship is between **Staff** and **Lab** node. The **Student** node models enrolled student in the system. It has two properties: sid and name. A student can enrol in multiple course offerings. This is modelled as **ENROLLED** relationship. A student can obtain a score from each course offering. The score is modelled as the property of the **ENROLLED** relationship. Each student, when enrolled in a course is also assigned in one of the labs. This is also modelled as **ENROLLED** relationship. A student can work as tutors in a lab, this is modelled as **TUTORS** relationship. Below is a sample graph showing a few nodes and their relationships.



1. [**3 points**] Write a query to find the average passing mark of each course offered in 2018. Here passing mark means a mark that is 50 or above.

```
MATCH (t: Timetable {year:2018})<-[r:ENROLLED]-(s:Student)
WHERE r.score >= 50
RETURN t, avg(r.score)
```

2. [**6 points**] The problem domain has many constraints for nodes and their relationships. Maintaining such constraint is largely the responsibility of developers. One option is to write queries to periodically check if there is any violation. This part asks you to develop queries to test the following constraints.

a) A student cannot be the tutor of any course he/she is also currently enrolled in as a student.

```
MATCH (t:Timetable)<-[:ENROLLED]-(s:Student) -[:TUTORS]-> (:Lab)<-[:LABSCHEDULE]-(t)
RETURN s
```

b) A student cannot enrol in more than one lab of the same course.

```
MATCH (t:Timetable)-[:LABSCHEDULE]->(:Lab)<-[:ENROLLED]-(s:Student)-[e:ENROLLED]->
(:Lab)<-[:LABSCHEDULE]-(t:Timetable)<-[:SCHEDULE]-(c:Course)
RETURN s
```

3. [**3 points**] We want to use the data to find candidate tutors of a given course. A candidate tutor is a student achieved HD (85 or above) in previous offerings of the same course. Now write a query to find candidate tutors for 2019 COMP5222 offerings.

```
MATCH (s:Student)-[r:ENROLLED]->(t:Timetable)<-[:SCHEDULE]-(c:Course{code:”COMP5222”)
WHERE t.year < 2019 and r.score >= 85
REUTRN s.name
```

4. [**4 points**] Assuming no node property index has been set. Describe the execution plan of the following query

```
MATCH (c:Course)-[]-(t:Timetable{year:2019})-[]-(:Lab)-[r:TUTORS]-(tutor)
RETURN c.title, labels(tutor), count(tutor)
```

Assumption that there are less Timetable rows than any other labels. Conduct COLSCAN to find rows with year = 2019. Then find courses that are have any relation to the current output. Then Labs then tutors associated.

Part 5 - 6 refer to the following nodes/relationships and their respective IDs. Assume the ID value indicates the creation order; smaller value means early creation. For instance, relationship **s** with id **0** is created before the relationship **ls** with id **1**.

Node or Relationship	**ID**
(c1 :Course {code: “COMP5111”, title: “C1” })	0
(t1 :Timetable {year:2019, semester:2, venue:LT110, time:“Tue18”})	1
(l1 :Lab{code:“T20A”; location:“SIT114”})	10
(st1 :Student{sid:1234; name: “Alice”})	20
(c1)-[s :SCHEDULE]->(t1)	0
(t1)-[ls :LABSCHEDULE]->(l1)	1
(st1)-[e1 :ENROLLED]->(t1)	2
(st1)-[e2 :ENROLLED]->(l1)	3

5. [**6 points**] Write down the content of the following byte ranges in the relationship record at byte offset 34:

- byte 1~4: firstNode 1
- byte 5~8: secondNode 10
- byte 13~16: first nodes previous relationship id 0
- byte 17~20: first nodes next relationship id. 2
- byte 21~24: second nodes previous relationship id null
- type 25~28 : second nodes next realtionsip id 3

6. [**3 points**] Which node has its record at byte offset 300? Which records(s) are included in this node’s doubly linked list of relationship records?

The node with its record stored at byte offset 300 is 20. The record for e1 relationship and e2 relationship are stored in node 20’s doubly linked list.

All parts of this question are based on a **Dynamo cluster** with five nodes: n_0, n_1, n_2, n_3 and n_4. Their corresponding tokens are shown on the following left hand side table. The ring space for consistency hashing is between 0~99. The cluster has a **replication factor** 3**.** The **preference list** contains **4 nodes**. The consistency configuration (N, R, W) of the system has the value (3,2,2). One of the tables stored in this cluster contains information about faculties in a university. The **faculty name** is used as **key**. Sample keys and their corresponding hash values in the ring space are given in the right hand side table.

Node	**Token**
n_0	5, 50
n_1	20, 85
n_2	35, 60
n_3	75
n_4	95

Key	**Hash value**
Arts	31
Business	93
Education	29
Engineering	13
Law	71
Medicine	47
Science	53

1. [**4 points**] What is the preference list of key "**Science**"?

   [n_2, n_3, n_1, n_4]

2. [**4 points**] Which node has the least number of keys? What are the keys on this node?

   n_4, has the least number of keys. It only stores Business and Law key.

Node	**Keys**
n_0	Arts, Business, Education, Engineering, Medicine (co-or)
n_1	Business, Engineering (co-or), Law, Science
n_2	Arts (co-or), Education (co-or), Engineering, Medicine, Science (co-or)
n_3	Arts, Education, Law (co-or), Medicine, Science
n_4	Business (co-or), Law

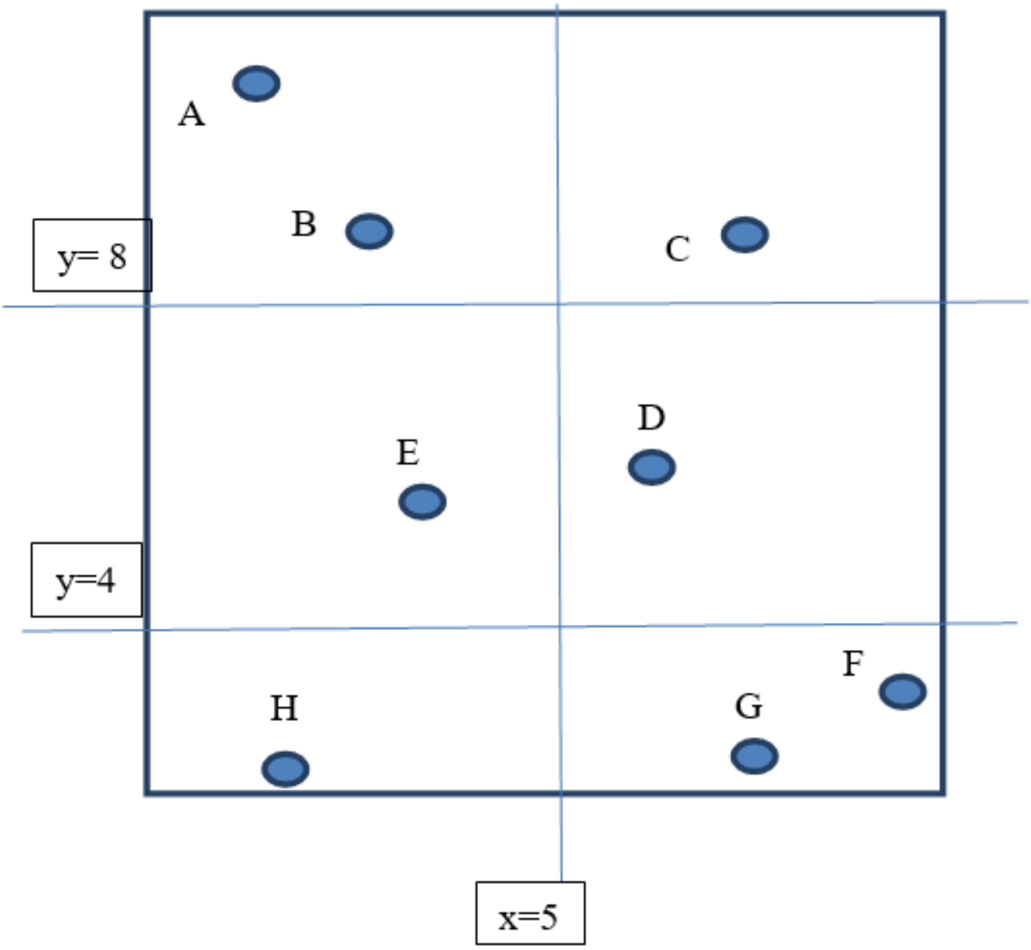
3. [**2 points]** Suppose all versions of the object with key "**Law**" have the same vector clock ([n_3, 10]), what do we know about the update history of this key?

   This means that all versions of the object with key "Law" has been updated 9 times after the first insert and all operations were coordinated by node n_3.

4. [**5 points**] Now suppose all other nodes except $\mathbf{n_3}$ are available during the next update of key "**Law"**, what would be the vector clock of the new version? Which nodes would have the new version?

If node n_3 is unavailable then the next node in the preference list for object with key “Law” will co-ordinate the write. The new vector clock of the new version would be [(n_3, 10), (n_1, 1)]. The preference list of object with key “Law” is [n_3, n_1, n_4, n_0]. The nodes that have the new version are [n_1, n_4, n_0, n_2]. n_2 has the new version as replication is carried over to the next node until n_3 is available. This is because the system has set the number of replication as 3.

Assume we have a collection of 2D points and our chosen indexing method segments the underlying space as follows. Name the indexing method used and show the index structure using the sample points in the figure. **[4 points]**



The indexing method used is the Space Segmenting method. A 2 dimension array for bucket location: assume the buckets are named as 1~6, the actual array is [[1,2],[3,4],[5,6]]
where bin 1 contains points [A, B],
bin 2 contains point C,
bin 3 contains point E,
bin 4 contains point D,
bin 5 contains point H,
bin 6 contains points[G, F,

The linear scales for grid line location: assume the range of x and y are both [0,10],
the scales are x:[0,5,10] o y:[0,4,8,10]
X: [0, 5, 10]
Y: [0, 4, 8, 10]

Nine-Intersection Model can be used to specify topological relationship of objects in 2D space. Assuming row represents object A, column represents object B. What topological relationship does the following nine-intersection model matrix represents: **[4 points]**

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

This nine-intersection model matrix represents that object A covers object B.

MBR is an important concept in spatial data model. Explain what is MBR and how it is used in spatial query. **[4 points]**

MBR is an acronym for Minimum Bounding Rectangle. It is used as the key for R-Tree indexing for spacial data. The interior nodes represents MBR of its children. Leaf node represents a number of MBR of the spatial objects in the database. MBR for different nodes may overlap. One MBR can be covered by many upper level node's MBR, but it can only be associated with one node.
Each internal node can have maximum 3 and minimum 2 children.

1. Describe the main features of Gorilla's data model.
   Designed for write heavy

2. Explain the main differences between Gorilla and InfluxDB in terms of how time series data are handled.
3. You are given two data sets. Each data set contains hourly data on the number of pedestrians that passed two different locations. Use a table to illustrate how you would model this data using InfluxDB.

Question 12 pts

Which one of the following is TRUE about shard in MongoDB?

Group of answer choices
Each shard is a replica of the other shard in the cluster
Each shard holds a continuous range of shard key values of a collection
Each shard can have more than one chunks of a collection
The primary shard is responsible for write operations in the shard

Flag this Question

Question 22 pts

Assuming a Bigtable tablet contains wide rows and the data about the same row are usually inserted/updated by multiple write operations. A read query looking for one such wide row would likely to assemble the result from

Group of answer choices
The log file and one or more SSTable files belonging to this tablet
The log file and memories of all tablet servers that have handled the write query of this row.
The memory of the tablet server serving this tablet and one or more SSTable files belonging to this tablet
The memory of the tablet server serving this tablet and the log file of this tablet

Flag this Question

Question 32 pts

Master-Slave Replication is a simple scale out option. Which one of the statements is TRUE about this option?

Group of answer choices
The slaves should keep a consistent state with the master at all times
The master should not receive any read request
The master should receive all write requests
There should be one master and two slave nodes

Flag this Question

Question 42 pts

Which one of the following is NOT TRUE about SSTable in Bigtable?

Group of answer choices
They are organized as table format
Data about one row may be stored in multiple SSTable files
They are immutable
They are created by memory flush or compaction

Flag this Question

Question 52 pts

Which one of the following is NOT TRUE about MongoDB aggregation stage?

Group of answer choices
The output document number of $project stage is always equal to its input document number
The output document number of $unwind stage may be greater than, equal to or less than its input document number
The output document number of $lookup stage is always less than or equal to its input document number.
The output document number of $group stage is always less than or equal to its input document number.

Flag this Question

Question 62 pts

Bigtable stores multiple versions of a column by design. This is achieved by

Group of answer choices
Adding a timestamp to each value
Adding a timestamp to each row
Adding a timestamp to each column
Adding a timestamp to each column family

Flag this Question

Question 72 pts

Which of the following is TRUE about ROOT and METADATA tablet in Bigtable?

Group of answer choices
All queries to METADATA tablets should go through master; client cannot query METADATA tablet directly.
The METADATA table may split and its tablets can be managed by different tablet servers
The ROOT tablet is stored in Chubby to ensure strong consistency and durability
The ROOT tablet stores information about the tablet servers, each row represents a tablet server.

Flag this Question

Question 82 pts

Which one of the following is NOT TRUE about Bigtable Architecture

Group of answer choices

There is only one master server in the cluster
A table may be split into many tablets and managed by different tablet servers
The master server is responsible for all write operations
A tablet server may manage tablets belonging to many different tables

Flag this Question

Question 92 pts

Which one of the following is NOT TRUE about Chubby service in Bigtable?

Group of answer choices
Chubby service ensures that there is at most one active master server at any time
Chubby service stores the root tablet location
Chubby service knows the list of tablet servers in Bigtable
Chubby service is contacted in every read/write operation

Flag this Question

Question 102 pts

MongoDB uses GeoJSON object to store spatial data. What spatial object(s) is(are) defined by the following GeoJSON object?

```
{type: "MultiPolygon",
  coordinates : [
    [ [ [ 0, 0], [ 3, 6], [ 6, 1], [ 0, 0]],
    [ [ 2, 2], [ 3, 3], [ 4, 2], [ 2, 2]]],
    [ [ [ 0, 0], [ 0, 6], [ 6, 6], [ 6,0],[0,0]] ]
 ]
}
```

Group of answer choices
A polygon with one triangle exterior and two holes: a triangle and a square
A collection of three polygons: two triangles and one square
A collection of two polygons: one triangle with a triangle hole and one square
A polygon with one square exterior and two triangle holes

Flag this Question

Question 11~16 is based on a MongoDB database **book_review**. The database contains two collections: **books** and **reviews.** The database is designed to keep information about books and their reviews. Each document in the **books** collection stores information about a particular book.
All **book** documents include the following fields: **_id**, **parent_id**, **title**, **author**, **publisher**, **edition**.
The **parent_id** field is used to associate different editions of the same book. If a book has published many editions, there will be one document for each edition, all of which will have the same **parent_id** value.
The **parent_id** value will be the first edition's **_id** value.   Each document in the **reviews** collection stores a review of a particular book. A **review** document has a unique **_id**, it also stores the **_id** of the book the review is about. It may include the review text, reviewer's name, time of the review, the rating given by the reviewer and helpful vote count of this review. Note that **review_time** is of **Date** type, for simplicity, the **Date** literal is written in string format.

The following indexes have been created:

```
        db.reviews.createIndex({rating:-1, review_time:-1, helpful_vote:-1,});
        db.reviews.createIndex({review_text: "text"})
        db.books.createIndex({parent_id:1})
        db.books.createIndex({title: 1})
```

Below are two sample documents: a **book** document and a **review** document of the book:

```
{
        _id: 5,
        parent_id: 5,
        title: “Sapiens: A Brief History of Humankind”,
        author: “Yuval Noah Harari”,
        publisher: “Harper Perennial”,
        edition: 1
}

{
        _id: 1,
        book_id: 5,
        reviewer: “Bill Gates”,
        review_text: “What’s unique about Harari’s take is that he focuses on the power
of stories and myths to bring people together”,
        rating: 5,
        review_time: “2016-05-17”,
        helpful_vote: 3109
}
```

Flag this Question

Question 112 pts

In which one of the following queries, index will not be used in the query plan?

Group of answer choices

```
db.books.find({parent_id: 1, editions: {$gt:1}})

db.reviews.find({rating: {$gt: 3} , helpful_vote: {$gt:100}})

db.reviews.find({review_text: {$regex: “^Sapien”} , helpful_vote: {$gt:100}})

db.books.find({title: {$regex: “^Sapien”},{edition:1})
```

Flag this Question

Question 122 pts

Assume an early query **books.find({publisher: "ABC"}).count()** returns 5. The client then issues an update query to change the name of the publisher and to add a field to store the location of the publisher.

```
db.books.update({publisher: "ABC"}, {$set: {publisher:  "ABC Inc.", Location: "NYC"}}
)
```

Which of the following is NOT TRUE?

Group of answer choices

A concurrent query **db.books.find({Location: "NYC"})** may return 1~5 documents with the following fields values: **{publisher: "ABC Inc", Location: "NYC"}**

A concurrent query **db.books.find({publisher: "ABC Inc."}).count()** may return any number between 0 and 5.

A concurrent query **db.books.find({publisher: "ABC"}).count()** may return any number between 0 and 5

A concurrent query **db.books.find({Location: "NYC"})** may return 1 ~5 documents with the following fields values: **{publisher: "ABC", Location: "NYC"}**

Flag this Question

Question 132 pts

Which one of the following queries cannot use index for sorting?

Group of answer choices

```
db.reviews.find({rating: 4).sort({review_time: 1})

db.reviews.find({rating: {$gte:4}, , helpful_vote: {$gte:100}}).sort({rating:-1})

db.reviews.find({rating: {$gte:4}}).sort({rating:1,review_time:1})

db.reviews.find({rating: 4, helpful_vote: {$gte:100}}).sort({helpful_vote:-1})
```

Flag this Question

Question 142 pts

What does the following query return?

```
db.reviews.find ({book_id: 3}, {review_text:1, rating:1}).sort({review_time: -1}).limit
(1)
```

Group of answer choices

The latest review of a book with text equals 1, rating equals 1 and id equals 3

The latest review text with rating equals 1 of a book with id equals 3

The latest review text and rating of a book with id equals 3

The query does not return anything because the syntax is wrong

Flag this Question

Question 15~16 are related with the following aggregation:

```
db.books.aggregate([
      {$match:{parent_id:1}},
      {$lookup:{
                  from: "reviews",
                  localField: _id,
                  foreignField: "book_id",
                  as: "reviews"
       }},
      {$project:{reviews:1, review_count:{$size: "$reviews"}}},
      {$match:{review_count:{$gte:1}}},
      {$unwind: "$reviews"},
      {$group:{_id:null, total_reviews:{$push:"$reviews"} }}
])
```

Flag this Question

Question 152 pts

Which fields are included in the output document of the **$project** stage of the aggregation?

Group of answer choices

_id, reviews, review_count
book_id, reviews, review_count
reviews, review_count
parent_id, title, author, publisher, edition, reviews, review_count

Flag this Question

Question 162 pts

Which stage of the following aggregation would not change the structure of input documents?

Group of answer choices
The $lookup stage
The $match stage
The $project stage
The $group stage

Flag this Question

Question 17-19 are related with the following scenario:

A **restaurants** collection containing the following 4 documents:

```
    {_id: 1, name: "Central Perk Cafe", violations: 3}
    {_id: 2, name: "Rock A Feller Bar and Grill", violations: 2}
    {_id: 3, name: "Empire State Sub", violations: 5}
    {_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

The collection is stored in a replica set with three members. All members have the same copy of the data at the beginning of the scenario. The following write query is sent to this collection:

```
db.restaurant.updateMany(
      { violations: { $gt: 4 } },
      { $set: { Review : true , reviewer: "R. Coltrane"} }
 )
```

The write was completed in primary at t_0. It was completed in secondary 1 at t_2 and completed in secondary 2 at t_3; The primary receives acknowledgement from secondary 1 in t_4; the secondary 1 receives notification from primary to update its write concern majority copy at t_5; the secondary 2 receives notification from primary to update its write concern majority copy at t_6. Note that subscript of time indicates order. For instance, t_0 is the first time and t_2 is before t_3.

The following concurrent read query also sent to the same collection. There is no other concurrent write.

```
db.restaurant.find({violations: { $gt: 4 }})
```

Flag this Question

Question 172 pts

Assume the read preference is set to *secondary* and the read concern is set to *majority*. Secondary 2 receives the read quest between t_4 and t_5. What would be the results of the query?

Group of answer choices

```
{_id: 3, name: "Empire State Sub", violations: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

```
{_id: 3, name: "Empire State Sub", violations: 5, Review: true, Reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8, Review: true, Reviewer: "R. Coltrane"}

{_id: 3, name: "Empire State Sub", violation: 5, Review: true, reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8}

{_id: 3, name: "Empire State Sub", violations: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8, Review: true, Reviewer: "R. Coltrane"}
```

Flag this Question

Question 182 pts

Assume the read preference is set to *primary*; and the read concern is set to *local*. Which of the followings could NOT be the results of the read query?

Group of answer choices

```
{_id: 3, name: "Empire State Sub", violation: 5, Review : true, Reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8 , Review : true, Reviewer: "R. Coltrane"}

{_id: 3, name: "Empire State Sub", violation: 5}
{_id: 4, name: "Pizza Rat's Pizzaria", violation: 8 , Review : true, Reviewer: "R. Coltrane"}

{_id: 4, name: "Pizza Rat's Pizzaria", violation : 8 , Review : true, Reviewer: "R. Coltrane" }

{_id: 3, name: "Empire State Sub", violations: 5, Review: true, Reviewer: "R. Coltrane"}
{_id: 4, name: "Pizza Rat's Pizzaria", violations: 8}
```

Flag this Question

Question 192 pts

Now assume the read preference is set to *primary*; and the read concern is set to *majority*. The primary receives the read request between t_4 and t_5, what could be the results of the read query?

Group of answer choices

```
{_id : 3, name : "Empire State Sub", violation : 5, Review : true, reviewer: "R. Coltrane" }
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 , Review : true, reviewer: "R. Coltrane"}

{_id: 3, name : "Empire State Sub", violation : 5}
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 }

{_id: 3, name : "Empire State Sub", violation : 5}
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8 , Review : true, reviewer: "R. Coltrane"}
```

```
{_id: 3, name : "Empire State Sub", violation : 5, Review : true, reviewer: "R. Coltrane" }
{_id: 4, name : "Pizza Rat's Pizzaria", violation : 8}
```