The allowed time for the exam this time will be 120 minutes. Be sure to pay attention to the time and to budget your exam time accordingly! (Also, don't forget to periodically save your answers.)

The exam is open pre-prepared cheat sheet, open book, open notes, open web browser, and even open data platforms. However, **you are not allowed to communicate** with or otherwise interact with any other students (or friends) during the course of the exam, and this includes your HW brainstorming buddy. This exam is to be **a solo effort!**

Read each question carefully, in its entirety, and then answer each part of the question.

If you don't understand something, make your best guess; if you find ambiguities in a question, note the interpretation that you are taking.

**Integrity Acknowledgement:** I certify that I am taking this exam myself, on my own, with honesty and integrity, without interaction with others during the exam, and without having obtained any information about the exam’s content from others prior to taking it.

☐ True

☐ False

---

**A Messy Example**
This being a database class, we'll need data for the exam! To that end, we'll make use of the following data - it's most succinctly viewed and understood in JSON form, so we'll start with a JSON version and introduce other versions as needed throughout the exam.

This data is from a hypothetical website, Messy.com, that was started to give tech users a place to vent to one another or just chat in these stressful times. Messy.com has a bit of an e-mail flavor and a bit of a LinkedIn flavor to it. Here is the data:

Users:

```json
[{
    "uid": 1,
    "alias": "Margo",
    "name": "Margarita Stoddard",
    "nickname": "Mags",
    "age": 29,
    "since": "2019-08-20",
    "friends": [2, 3, 6],
    "jobs": [{
        "org": "ESRI",
        "started": "2018-06-17",
        "ended": "2018-12-31"
    },
    {
        "org": "Code tech",
        "started": "2018-01-02"
    }],
    "gender": "F"
}
,
{
    "uid": 2,
    "alias": "Isabel",
    "name": "Isabel Dunn",
    "nickname": "Izzy",
    "age": 26,
    "since": "2020-01-22",
    "friends": [1, 4],
    "jobs": [{
        "org": "HA Labs",
        "started": "2018-04-27"
    }]
}
, {
    "uid": 3,
    "alias": "Emory",
    "name": "Emory Unk",
    "age": 35,
    "since": "2018-07-10",
    "friends": [1, 5],
    "jobs": [{
        "org": "ESRI",
        "started": "2018-06-17",
        "ended": "2018-01-26"
    }]
}
, {
    "uid": 4,
    "alias": "Nick",
    "name": "Nicholas Stroh",
    "age": 29,
    "since": "2018-12-27",
    "friends": [2],
    "jobs": [{
        "org": "HA Labs",
        "started": "2018-06-08"
    }],
    "gender": "M"
}
, {
    "uid": 5,
    "alias": "Von",
    "name": "Von Kemble",
    "age": 40,
    "since": "2018-01-05",
    "friends": [3, 6],
    "jobs": [{
        "org": "Code tech",
        "started": "2018-11-27"
    }],
    "gender": "F"
}
, {
    "uid": 6,
    "alias": "Willie",
    "name": "Willis Wynne",
    "nickname": "W",
    "age": 35,
    "since": "2017-01-17",
    "friends": [1, 3],
    "jobs": [{
        "org": "IVM",
        "started": "2009-05-15"
    }]
}]
```

Messages:

```json
[{
    "mid": 1,
    "from": 3,
    "to": [1, 2],
    "sent": "2020-12-16",
    "loc": "San Jose",
    "msg": "Here visiting Goggle"
}
, {
    "mid": 2,
    "from": 1,
    "to": [3],
    "sent": "2020-12-23",
    "replyto": 1,
    "loc": "San Diego",
    "msg": "Currently visiting Terrydata"
}
, {
    "mid": 3,
    "from": 2,
    "to": [1],
    "sent": "2020-12-25",
    "loc": "Laguna",
    "msg": "The ocean view here rocks"
}
, {
    "mid": 4,
    "from": 1,
    "to": [3, 4, 5],
    "sent": "2020-12-31",
    "loc": "Point Loma",
    "msg": "It’s foggy today"
}
, {
    "mid": 5,
    "from": 5,
    "to": [1, 2, 3, 4, 6],
    "sent": "2021-01-16",
    "loc": "Sacramento",
    "msg": "Feels like capitol punishment :-("
}
, {
    "mid": 6,
    "from": 2,
    "to": [3],
    "sent": "2021-01-31",
    "replyto": 1,
    "loc": "Newport Beach",
    "msg": "Checking out Goggle Irvine"
}
, {
    "mid": 7,
    "from": 5,
    "to": [1, 2, 3, 4, 6],
    "sent": "2021-02-07",
    "loc": "Laguna",
    "msg": "Great restaurants here"
}
, {
    "mid": 8,
    "from": 1,
    "to": [3],
    "sent": "2021-02-14",
    "loc": "La Jolla",
    "msg": "UCSD has a big campus"
}
, {
    "mid": 9,
    "from": 1,
    "to": [3],
    "sent": "2021-02-26",
    "replyto": 1,
    "loc": "San Jose",
    "msg": "Housing is gonna be pricey"
}
, {
    "mid": 10,
    "from": 1,
    "to": [2],
    "sent": "2021-03-15",
    "replyto": 3,
    "loc": "La Jolla",
    "msg": "The ocean view rocks here too"
}
, {
    "mid": 11,
    "from": 4,
    "to": [2],
    "sent": "2021-03-15",
    "loc": "Seattle",
    "msg": "It’s raining again here"
}
, {
    "mid": 12,
    "from": 6,
    "to": [4],
    "sent": "2021-03-15",
    "loc": "Alameda",
    "msg": "The traffic here sucks"
}
, {
    "mid": 13,
    "from": 6,
    "to": [4],
    "sent": "2021-04-02",
    "loc": "Alameda",
    "msg": "Just passed the Raiders old stadium"
}
, {
    "mid": 14,
    "from": 2,
    "to": [1, 3],
    "sent": "2021-05-01",
    "replyto": 2,
    "loc": "Newport Beach",
    "msg": "Checking out Expellian"
}
, {
    "mid": 15,
    "from": 5,
    "to": [1, 2, 3, 4, 6],
    "sent": "2021-05-01",
    "replyto": 2,
    "msg": "Interviewing with a stealth startup"
}]
```

**Users** have a user id, a unique alias that they go by, a name, an age, a date when they joined the site, and a job history saying where they've worked and from when to when. (If they still work for an organization, the associated ended date will be missing.) Users can also add other attributes if they wish, and a few of them have done so in the data shown.
User of the Messy.com website and its associated app can send messages to one or more other users. **Messages** have a message id, info on who they are sent from and sent to, info on when and where they were sent, and the message body itself. Some messages may be sent as a response to (i.e., a reply to) another message.

**NOTE**: You shouldn't need to load this data into anything for this exam! You will hopefully be able to look at each question and the sample data and answer based on your experience and knowledge. The intent is for you to think, applying the principles that you've learned, and not to use the different systems in real-time to answer the questions. (You may want to pdf-print or screen-shot the example data so that you can refer to it in another window during the exam. Do not open the exam itself twice in different windows, as that could end badly if you do an accidental "save" in the wrong window!)

**Q2 Legal Briefs**
19 Points

Choose the best alternative from among the offered answers to each of the following questions.

**Q2.1**
1 Point

Is the following a legal JSON value?

```
{"name":"E.Z.",age: 21,[1, 2, 3]}
```

- ☐ Yes
- ☐ No

**Q2.2**
2 Points

Consider the following JSON object. Suppose that Couchbase Server were used to store the sample JSON data from our running example in a bucket
called Messy, with a "kind" field being added and used to discriminate between the different kinds of objects in the bucket, and with the Couchbase Analytics Service being set up to route its copy of the data to Users and Messages datasets based on the type field – just like you did in the Couchbase Server homework. Would Couchbase Analytics allow the insertion of this object into its Users dataset if it were routed there after insertion into the Messy bucket in the Data service?


- Yes
- No

Q2.3
2 Points

Consider the JSON version of our Messy example. Suppose that the real data is several TB in size, including 500GB of Message records, and that the following N1QL for Analytics query must be run frequently:

```sql
SELECT m.`from`, COUNT(*) AS numsent, AVG(ARRAY_COUNT(m.`to`)) AS avgto FROM Messages m GROUP BY m.`from` ORDER BY m.`from`;
```

Suppose the Couchbase cluster currently has 10 nodes, with the Analytics Service deployed on 2 of the 10 nodes and the Data and Query and Index Services deployed on the remaining 8 cluster nodes. (Let's hear it for multidimensional scaling!) If this analytical query is currently running too slowly by a factor of 4, and the number of messages is also expected to double by year-end, how many nodes in total should the overall cluster aim to have in order to meet the company's storage and speed requirements for this data analysis at the end of the year?
Q2.4
2 Points

Many parallel RDBMSs use two-step parallel hash-based algorithms to process large group-by aggregate queries. What do you expect to happen when the nature of the input is such that each node's share of the input data is larger than the amount of main memory available on a node, e.g., if there are hundreds of millions of small groups, as there might well be for the query above?

- the system will crash with an out-of-memory error
- the query will run more slowly
- the hash-based approach won't run, so a different algorithm will be used

Q2.5
2 Points

Suppose a parallel RDBMS uses range-based partitioning on each table's primary key to distribute its data to the nodes in the cluster. Consider a relational version of our Messy data. Assuming that user names are mostly unique, but that the Users table's primary key is user id, which of the following indexing choices would likely be best for a query like the following SQL query:

```
SELECT * FROM Users u WHERE u.name = 'Vince Guaraldi';
```
a global secondary index on Users(name)

- a local secondary index on Users(name)
- no index

**Q2.6**
2 Points

And which indexing choice would likely be best for a SQL query like:

```
SELECT * FROM Messages m WHERE m.msg LIKE '%Fauci%';
```

- a global secondary index on Messages(msg)
- a local secondary index on Messages(msg)
- no index

**Q2.7**
2 Points

Consider storing the Messy data in a NoSQL system that uses a peer-to-peer (P2P) replication approach to consistency. Assuming N copies of each data item, any values of R (the number of copies to read) and W (the number of copies to write before acknowledging the write) will suffice to maintain data consistency and deliver consistent reads as long as R + W > N.

- True
- False

**Q2.8**
2 Points

Redis is most accurately *technically* categorized as a:
Q2.9  
2 Points

Look closely at the nature of the Users data in our Messy example. Consider normalizing the data for storage in a parallel RDBMS. What would be the smallest number of tables needed to hold the User data in a well-designed relational schema? Assume the goal is a 1NF schema (no cheating :-) -- and in fact a one-fact, one-place design (i.e., a clean translation from its E-R form to a relational schema).

- 1
- 2
- 3
- 4

Q2.10  
2 Points

Based on good design patterns and anti-patterns for NoSQL database design, and what you know (common sense) about the world, consider the Messy website's design decision to nest Users' job history information in its User objects. Is this a reasonable design decision, or was it a bad choice?

- this particular nesting is a reasonable choice
- this particular nesting was a bad choice
Q3 No Shortage of NoSQL to Know About
30 Points

Over the course of the quarter you have learned about a wide range of technologies, some more in-depth than others. (Some of them were just touched upon early in the class, or in the readings, or in the last wrap-up lecture's tour of "everything else." This problem will test the breadth of your knowledge retention -- match each statement with the platform or technology that is the best fit, being careful not to choose any of them twice in the event that you feel like there may be multiple potential fits. (This is a matching problem, so you should use each answer at most one time...!) Your list of choices is:

(A) Parallel RDBMS
(D) MongoDB
(G) Distributed file system
(J) Data streaming system
(M) Data lake

(B) Key-Value store
(E) Couchbase Server
(H) Map/Reduce
(K) Kafka
(N) Elasticsearch

(C) Cassandra
(F) Neo4J
(I) Spark
(L) Column store
(O) XML

database

Next to each statement below you are to enter just the parenthesized uppercase letter for your choice (i.e., do not forget the ()'s or fail to use uppercase or all of your answers will be auto-graded incorrectly). For example:

This is where the builders of ancient Greek and Roman temples would often shop.

**(L)**

Q3.1
2 Points

Like an upside-down database: data is transient while queries are persistent.

**(J)**

Q3.2
2 Points
Like a sideways database: tables are stored vertically and compressed rather than being stored horizontally (i.e., row-wise).

Q3.3
2 Points

Prides itself for laziness when interacting with data scientists.

Q3.4
2 Points

Hash-partitioned data and partitioned-parallel query execution techniques originated here.

Q3.5
2 Points

No schema required, with query capabilities intended mainly to facilitate single-collection queries and analyses.

Q3.6
2 Points

Favors availability over consistency and scales to large geographical clusters.
Q3.7
2 Points

A platform of choice for querying large volumes of textual documents.

(N)

Q3.8
2 Points

Essentially got run over technically by JSON-based document stores.

(O)

Q3.9
2 Points

No schema required, with query capabilities based on generalizing/relaxing SQL.

(E)

Q3.10
2 Points

Provides scalability and single-object operations with the world’s simplest API.

(B)

Q3.11
2 Points

Repository for Big Data analytics platforms that do not store or index their own data.

(G)
Q3.12
2 Points

Large-scale storage for data collections, with schema-on-read rather than tightly managed schema-driven schema-on-write.

(M)

Q3.13
2 Points

No schema required, focuses on use cases with flexible entities where relationships are paramount.

(F)

Q3.14
2 Points

Used to reliably move data in an enterprise, including support for topic-based messaging and both pub/sub and queueing consumption models.

(K)

Q3.15
2 Points

Its developers just provide 2-3 functions and then the system does the rest (including parallelization and fault-tolerant execution).

(H)

Q4 Cassandra of Troy
10 Points
Consider setting up a basic flat (essentially relational) version of the Messy schema for use on Cassandra:

```
CREATE TABLE Users (uid int, alias text, name text, nickname text, age int, since date, gender text, PRIMARY KEY (uid));
CREATE TABLE Friends (uid int, frid int, PRIMARY KEY ((uid), frid));
CREATE TABLE Jobs (uid int, org text, started date, ended date, PRIMARY KEY (uid, org));
CREATE TABLE Messages (mid int, sent date, replyto int, loc text, msg text, PRIMARY KEY (mid));
CREATE TABLE FromTo (mid int, "from" int, "to" int, PRIMARY KEY ((mid, "from"), "to"));
```

You should be able to envision the contents of these 1NF tables by looking at the JSON version of the data and then knowing how the normalization (flattening) process will distribute its content to the various Cassandra tables.

Let's consider how Cassandra will handle some CQL queries, given the keys specified above, as well as contemplating other possible key and table designs to facilitate them.

**Q4.1**
2 Points

What is the current situation for the following query:

```
SELECT uid, started, ended FROM Jobs WHERE org = 'ESRI';
```

- ☐ this query will run "as is"
- ☑ this query will run iff "ALLOW FILTERING" is added
- ☐ this query will not run with this table design
- ☐ this query will not run at all in Cassandra

**Q4.2**
2 Points
Which of the following key choices would work and enable this query to run "as is"? (Pick as many as appropriate; look closely at the key designs in doing so.)

- PRIMARY KEY ([org])
- PRIMARY KEY ([org, uid])
- PRIMARY KEY ([org], started, uid)

Q4.3
0 Points
(Blank - ignore.)

Q4.4
0 Points
(Blank - ignore.)

Q4.5
1 Point
What is the current situation for the following query:

```
SELECT u.alias, u.name, j.org, j.started
FROM Users u, Jobs j
WHERE u.uid = j.uid AND j.org = 'ESRI'
ORDER BY j.started;
```

- this query will run "as is"
- this query will run iff "ALLOW FILTERING" is added
- this query will not run with this table design
- this query will not run at all in Cassandra
Q4.6
5 Points

The query in the previous question turns out to be one of the most important queries that Messy users want answered (with the organization name being a parameter on a web page), as they like to pick one another's brains about current and past employers. Show what you could add to the Messy database to support the answering of this query efficiently. Your explanation should be given in the form of a CREATE TABLE statement. (Note: Be sure to include an appropriate PRIMARY KEY clause in your answer!)

CREATE TABLE UserJobInfo (  
    alias text,  
    name text,  
    org text,  
    started date,  
    PRIMARY KEY ((org), started, alias)  
);

Q5 Time for a Document-ary
20 Points

It's time to make sure that your MongoDB knowledge hasn't prematurely lapsed and that your Couchbase Server knowledge is sound -- which means that the JSON version of our Messy example will once again be the focus. Let's assume that the data has been loaded into two collections, Users and Messages, in MongoDB. Assume that in Couchbase Server the data resides in one Data Service bucket called Messy -- with each object also having a "kind" field (tagging Users and Messages as what they are) as well as being in two appropriately named Couchbase Analytics datasets. (Refer back to problem Q2.2 if needed).
Q5.1
5 Points

Consider the following pymongo MQL snippet:

```python
match_stage = {"$match": {"age": {"$lt": 40}}}
unwind_stage = {"$unwind": "$jobs"}
group_stage = {"$group": {"_id": "$jobs.org", "orgcount": {"$sum": 1 }}}
renaming_stage = {"$addFields": {"orgname": "$_id"}}
project_stage = {"$project": {"_id": 0}}
sort_stage = {"$sort": {"orgcount": -1}}
pipeline = [ match_stage, unwind_stage, group_stage, renaming_stage, project_stage, sort_stage ]
cursor = db.users.aggregate( pipeline )
print(dumps(cursor, indent=2))
```

Show the JSON output that would be produced if this pipeline were executed by Atlas against the Messy Users collection:

```
[ "orgcount": 2, "orgname": "HA Labs"],
```
Q5.2
5 Points
Consider again the above pymongo MQL snippet. Show the N1QL query that, if it were run against the N1QL for Analytics datasets, would produce the same output:

```
SELECT j.org, COUNT(*) AS orgcount
FROM Users u, u.jobs j
WHERE u.age < 40
GROUP BY j.org
ORDER BY orgcount DESC;
```

Q5.3
2 Points
Consider the following N1QL for Analytics query:

```
SELECT year, m.loc AS origin, COUNT(*) AS received
FROM Messages m, m.`to` dest
LET year = get_year(m.sent)
WHERE m.loc LIKE 'S%'
GROUP BY ROLLUP(year, m.loc)
ORDER BY year, m.loc;
```

Explain briefly in English roughly what this analytical query does:

It produces a report that shows the number of messages received in each year from each location. It also rolls up these numbers by year as well as overall.
Q5.4
8 Points

Show the output that would be produced if the aforementioned N1QL query were executed by Couchbase Analytics against the Messy Messages dataset. (Minor note: NULLs sort low by default in N1QL for Analytics, and the default sort order for ORDER BY is ASC.) If you want to save typing, you are welcome to show the result in a more tabular/CSV-like format -- with an appropriate header on top and then just the field values in the subsequent lines -- so that you don't have to type the resulting field names again and again.

```
[
  { "year": null, "received": 10, "origin": null },
  { "year": 2020, "received": 3, "origin": null },
  { "year": 2020, "received": 1, "origin": "San Diego" },
  { "year": 2020, "received": 2, "origin": "San Jose" },
  { "year": 2021, "received": 7, "origin": null },
  { "year": 2021, "received": 5, "origin": "Sacramento" },
  { "year": 2021, "received": 1, "origin": "San Jose" },
  { "year": 2021, "received": 1, "origin": "Seattle" }
]
```

Q6 Neo4J: How Much Do You Node?
10 Points

Now consider representing the Messy data in Neo4J using nodes and relationships. One approach to doing so would yield a graph database where the friends connections between User would look as follows:
If you compare this picture to the Messy JSON data, it should be clear how the directional FRIEND relationship encodes the friendship data. The following apoc.meta.graph summary, with some sample node data pasted in at the bottom of the picture, may also help you see how the node properties encode the values in the data.

Q6.1
4 Points

As a warmup, write a Cypher query to return the aliases and the ages of Users who have chosen to disclose their gender as female ('F').

MATCH (u:User) WHERE u.gender = 'F' RETURN u.alias, u.age;

Q6.2
6 Points

In the Messy world, if a user U1 puts another user U2 into their friends list, we say that U1 likes U2.

Write a Cypher query to return the aliases for those users who like someone who doesn't like them back.

MATCH (u:User)-[:FRIEND]->(f:User) WHERE NOT (f:User)-[:FRIEND]->(u:User) RETURN u
Q7 Still Have a Spark Left?

10 Points

Last but not least, suppose the Messy JSON data is now loaded into two Spark dataframes, Users_df and Messages_df, with associated SparkSQL table names of Users and Messages.

Q7.1

6 Points

What will the following SparkSQL query print, when run on our Messy data? Feel free to use a CSV-like answer format to minimize typing.

```sql
SELECT u.alias, u.name, job.org
FROM Users AS u LATERAL VIEW explode(u.jobs) AS job
WHERE job.ended IS NULL
ORDER BY job.org;
```

alias,name,org
Margo,Margarita Stoddard,Codetech
Von,Von Kemble,Codetech
Isbel,Isbel Dull,HA Labs
Nick,Nicholas Stroh,HA Labs
Willis,Willis Wynne,IVM

Q7.2

2 Points

Will running the following PySpark dataframe statement produce the same resulting data as the previous query?

```python
display(Users_df.withColumn("job", explode(Users_df.jobs))
    .select("alias", "name", "job.org").sort("job.org"))
```

- Yes
- No
The Pandas dataframe library in Python is popular with data scientists, and the Spark dataframe facility is intended to make them feel at home dealing with Big(ger) Data. Which if any of the following statements about the two dataframe facilities are true?

- Pandas uses lazy evaluation to avoid reading data into memory unnecessarily
- Spark employs partitioned parallelism to execute queries against large data sets
- SparkSQL query plans are usually different from Spark dataframe query plans
- Spark dataframe statements are 100% compatible with Pandas dataframe statements
Legal Briefs

2.1 (no title) 19 pts
2.2 (no title) 1 pt
2.3 (no title) 2 pts
2.4 (no title) 2 pts
2.5 (no title) 2 pts
2.6 (no title) 2 pts
2.7 (no title) 2 pts
2.8 (no title) 2 pts
2.9 (no title) 2 pts
2.10 (no title) 2 pts

QUESTION 3
NoShortage of NoSQL to KnowAbout 30 pts

3.1 (no title) 2 pts
3.2 (no title) 2 pts
3.3 (no title) 2 pts
3.4 (no title) 2 pts
3.5 (no title) 2 pts
3.6 (no title) 2 pts
3.7 (no title) 2 pts
3.8 (no title) 2 pts
3.9 (no title) 2 pts
3.10 (no title) 2 pts
3.11 (no title) 2 pts
3.12 (no title) 2 pts
3.13 (no title) 2 pts
3.14 (no title) 2 pts
3.15 (no title) 2 pts

QUESTION 4
Cassandra of Troy 10 pts

4.1 (no title) 2 pts
<table>
<thead>
<tr>
<th>Question</th>
<th>Title</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>(no title)</td>
<td>2 pts</td>
</tr>
<tr>
<td>4.3</td>
<td>(no title)</td>
<td>0 pts</td>
</tr>
<tr>
<td>4.4</td>
<td>(no title)</td>
<td>0 pts</td>
</tr>
<tr>
<td>4.5</td>
<td>(no title)</td>
<td>1 pt</td>
</tr>
<tr>
<td>4.6</td>
<td>(no title)</td>
<td>5 pts</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td><strong>Time for a Document-ary</strong></td>
<td><strong>20 pts</strong></td>
</tr>
<tr>
<td>5.1</td>
<td>(no title)</td>
<td>5 pts</td>
</tr>
<tr>
<td>5.2</td>
<td>(no title)</td>
<td>5 pts</td>
</tr>
<tr>
<td>5.3</td>
<td>(no title)</td>
<td>2 pts</td>
</tr>
<tr>
<td>5.4</td>
<td>(no title)</td>
<td>8 pts</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>Neo4J: How Much Do You Node?</strong></td>
<td><strong>10 pts</strong></td>
</tr>
<tr>
<td>6.1</td>
<td>(no title)</td>
<td>4 pts</td>
</tr>
<tr>
<td>6.2</td>
<td>(no title)</td>
<td>6 pts</td>
</tr>
<tr>
<td><strong>7</strong></td>
<td><strong>Still Have a Spark Left?</strong></td>
<td><strong>10 pts</strong></td>
</tr>
<tr>
<td>7.1</td>
<td>(no title)</td>
<td>6 pts</td>
</tr>
<tr>
<td>7.2</td>
<td>(no title)</td>
<td>2 pts</td>
</tr>
<tr>
<td>7.3</td>
<td>(no title)</td>
<td>2 pts</td>
</tr>
</tbody>
</table>