1. Data Partitioning (multiple answers)

1. ☐ does not have caveats
2. ☐ The main reason is scalability
3. ☐ means breaking a large dataset down into smaller ones
4. ☐ works well with network partitioning

2. Data Replication (multiple answers)

1. ☐ increases availability
2. ☐ makes the system fault tolerant
3. ☐ works well with changes in data
4. ☐ simplifies consistency

3. What are the basic elements of a data pipeline (independently from the frameworks)

1. Consumer, Producer, nodes
2. Pipes, Transfers, Input, Output
3. Pipes, Filters, Sources, Sinks

4. What are the basic operations of a data pipeline (independently from the frameworks)

1. transform, extract, move
2. persist, load, change
3. store, access, extract, transform, join/merge

5. What did change in the way we engineer data pipelines with the advent of Big Data?

6. In what part of the logical data pipeline Redis/MongoDB/NeoJ/Airflow can be useful?

- Draw the logical data pipeline, describe each part briefly
- Position system X, and explain

7. Missing Data are handled by ---------- (multiple answers)

1. ☐ Ignore the tuple
2. ☐ Building a machine learning model to predict the missing value based on other features in the data set
3. ☐ Fill in the missing value manually
4. ☐ Averaging over present values in other data points (samples) in the data set

8. Noisy data are handled by ---------- (multiple answers)

1. ☐ Regression
2. ☐ Binning
3. ☐ Clustering
4. ☐ Filling

9. Measures for data quality includes ---------- (multiple answers)

1. ☐ Accuracy
2. ☐ Completeness
3. ☐ Consistency
4. ☐ Partition Tolerance

10. A collection and a document in MongoDB are equivalent to .... in a relational database

1. Column and Row
2. Database and Table
3. Table and Column
4. Database and Column
5. Table and Row

11. What does MongoDB **NOT** provide?

1. ☐ schema
2. ☐ aggregates
3. ☐ joins
4. ☐ ACID transactions
5. ☐ projections

12. Which of the following MongoDB query is equivalent to the following SQL query:

```
SELECT COUNT(*)
FROM customer
```

1. db.customer.find().sum()
2. db.customer.find().AVG()
3. db.customer.find().count()
4. db.customer.find({All}).count()

13. Which of the following MongoDB queries is equivalent to the following SQL query

```
SELECT *
FROM customer_tbl
```

1. db.customer.get({All})
2. db.customer.find({})
3. db.customer.find({*})
4. db.customer.$find({*})

14. Redis can be used in

1. as a durable alternative to a relational database
2. for messaging and Pub/Sub, expiring /vanishing data apps, and caching
3. just for caching
4. as a transient alternative to mongoDB

15. In Redis, in order to get the score of the player with the name "Jane" in the Sorted-Set “players”

1. $GET score “Jane”
2. HSCORE "Jane"
3. ZINCRBY “Jane” 10
4. ZSCore players "Jane"

16. The Redis command to put this Key-Value pair in the DB (city:”Lyon”)

1. GET city “Lyon”
2. $INCR city
3. FLUSH city
4. SET city “Lyon”

17. What Redis data structure is best suitable for leaderboards and voting applications:

1. STRING
2. SET
3. SORTED SETS
4. BitMaps

18. What are the advantages of Graph Databases vs the other ones? (2-3 lines)

19. Given this Cypher statement, select the answer that best describes what data is returned from the query?

```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m:Movie)
RETURN m.title, m.released
SKIP 20 LIMIT 10
```

1. Titles and release years of movies acted by “Tom Hanks”, skipping the first 20 results and get the next following 10 results.
2. Movie nodes acted by “Tom Hanks”, skipping the first 10 results and get the next following 20 results
3. Titles and release years of movies acted by “Tom Hanks”, skipping the first 10 results and get the next following 20 results.
4. Count of movies acted by “Tom Hanks”

20. What operators does the Cypher query language share with SQL?

1. pattern matching, projection, join
2. projection, selection
3. where, select, from
4. none

21. Explain the CAP Theorem and draw the Venn-Diagram representation to position the various systems above (MongoDB, Redis, Neo4j) . Explain your answer.

22. The following airflow pipeline snipped identifies a ---- relation

```
node_1 >> [node_2, node_3, node_4]
```

1. many to one
2. many to many
3. one to one
4. one to many

23. the following airflow pipeline snipped identifies a ---- relation

```
[node_1, node_2, node_3] >> node_4
```

1. many to one
2. many to many
3. one to one
4. one to many

24. What are **Apache** Airflow's four main components?

1. Control Plane, Node, Scheduler and Container
2. Scheduler, Executor, Webserver and Database
3. Rest API, Cluster Manager, Airflow Client and Master Server
4. Metadata, Feature Store, Pipelines and Central Dashboard

25. What is a **DAG** in **Apache Airflow** ?

1. Directionally Active Graph, a representation of the Airflow scheduler.
2. Directed Acyclic Graph, a schematic of the workflow where tasks are edges and nodes represent direction and trigger rules.
3. Director Action Graph, the visual representation of how the control-flow is implemented
4. Directed Active Graph, the schema of the remaining operations to execute.

26. What is an **Operator** in **Apache Airflow**?

1. A unite of computation in the control plane represented as a node in the DAG
2. Symbols that represent functions(depending on the language)
3. A unite of datum in the data plan represented as an edge in the DAG
4. Type of task to execute e.g Celery/Local/Sequential

27. The following graph represents a data pipeline where:

1. A cluster is spawned
2. A query is made in order to pull some data from a database
3. A decision is made about how much processing this data needs
4. Processing happens
5. The cluster is shut down

28. Which operator should the task *make_decision* have in order to make sure that depending on the result of its upstream task, one of its two downstream ones will be skipped?

1. BranchPythonOperator
2. PostgresOperator
3. BashOperator
4. BaseHook

29. Which trigger rule should *shut_down_cluster* have such that irrespective of its upstream tasks' termination it will always be executed?

1. all_failed
2. none_skipped
3. all_done
4. none_failed