Junior Data Engineer Challenge

McqBigData1

tags: data-scientist, core, easy, advanced, data-engineer, big-data, mcq, 20-minutes, recruitment, full, data-science

A set of easy questions about Big Data.

Correct answer

1.Which of the following are the advantages of the ‘schema on read’ approach over ‘schema on write’?

A) Support for unstructured data

B) Faster loads to the storage layer

C) The flexibility of how data is consumed

D) Faster reads from the storage layer

2. When it comes to big data tools, what does the acronym YARN stand for?

A) Yet Another Resource Network

B) Yet Another Release Note

C) Yet Another Routing Network

D) Yet Another Resource Negotiator

3. You are trying to decide whether to use a single machine or cluster computing tools in your next project. Which of the following is the premise for using single machine architecture?

A) Your load might increase drastically over time.

B) You are only expecting to be loading a small amount of data.

C) You are expecting your tasks to be very memory-intensive.

4. Which of the following are useful Python packages for data processing and analysis projects?

A) Antigravity

B) Pandas

C) Seaborn

D) Pyglet

5. Your system is getting more traction and starts to require more computing power. Which of the following are reasons for scaling your system horizontally as opposed to vertically?

A) You are looking for more computing flexibility.

B) You are concerned about downtime when upgrading your machine.

C) You are unable to split your app into smaller logical blocks.

D) You want stable costs.

6. Which phase is usually the one we would like to get rid of, but might also be the most memory-intensive?

A) Map

B) Shuffle

C) Reduce

7. Which of the following statements about ELT are true?

A) An ELT model enables faster loading times than ETL.

B) An ELT model is an alternative to ETL.

C) With an ELT model, users can run transformations directly on the raw data.

D) An ELT model increases the time data spends in transit.

8. Which of the following are equivalent to AWS S3?

A) Google Big Query

B) Azure Blob Storage

C) Google Cloud Storage

D) Azure Data Factory

9. In terms of a Hadoop cluster, what is the heartbeat?

A) It is a signal sent from a name node to data nodes informing them about cluster health.

B) It is a signal sent from a name node to external applications informing them about cluster health.

C) It is a signal sent from external applications to a name node asking about system health.

D) It is a signal sent from data nodes to a name node informing it about node health.

10. Match the following technologies with their application:
1. Spark, 2. Cassandra, 3. Zookeeper, 4. Kafka, 5. Keras, 6. Superset
A. Database, B. Visualization, C. Orchestration, D. Analytics, E. Machine Learning, F. Streaming

A)1F, 2A, 3C, 4A, 5B, 6E

B) 1D, 2A, 3C, 4F, 5E, 6B

C) 1D, 2F, 3E, 4A, 5C, 6B

D) 1B, 2A, 3D, 4F, 5E, 6C

tags: python, data-engineer, apache-spark, pyspark, core, data-scientist, advanced, recruitment, data-science, real-life, 30-minutes, full, easy

Implement a method that will group and sort data using PySpark.

Task description

Group and sort data using PySpark.

Requirements

You are given a path to a file of comma-separated values (CSV), `jobs.csv`, which contains people's names and job titles, such as Dancer, Nurse, Pilot, etc. The dataset has two columns: 'name' (a string data type) and 'job' (also a string data type).

name	job
Tony Sullivan	Office manager
Mary Henry	Film editor
...	...
Tiffany Young	Dancer

Implement a `group_sort(input_path)` method that reads data from the `jobs.csv` file and returns a dictionary in which the keys are jobs and the values are counts of how many times each job appears within the dataset. The dictionary should be ordered by count (in ascending order), then job (in ascending order from A to Z). The `group_sort(input_path)` method takes one argument: `input_path` – a path to the CSV file containing the data.

Available packages/libraries

- Python 3.8 and all of its built-in packages
- Spark version 3.1.1

Hints

You can use `reduceByKey` and `sortByKey` operations on a key/value RDD object, or you can use `pyspark.sql` functions.

Examples

Calling the `group_sort(input_path)` method should return a dictionary with the following structure:

```
{'Job_title_1': count_job_1, 'Job_title_2': count_job_2, ..., 'Job_title_3': count_job_3}
```

If you would like to access CSV data sets locally you can download zipped files.

SqlLog

Given the system logs, return a list of users who logged in after a given timestamp.

A company was attacked by hackers at `2020-02-20 11:22:00`. The company wants to warn every user who logged into the system after this time. Your task is to find all such users.

You are given a table `logs` with the following structure:

```
create table logs (
      user_id integer not null,
      login_time timestamp not null
);
```

Each row of the table contains information about a single login: the id of a user (`user_id`) and the time when this user logged into the system (`login_time`). Note that a particular user could log into the system multiple times.

Write an SQL query that returns a table containing one column, `user_id`. Each row should contain the id of a user who logged in at least once strictly after `2020-02-20 11:22:00`. The result table should be sorted by `user_id` in increasing order.

Examples:

1. Given table:

```
+---------+---------------------+
| user_id |          login_time |
+---------+---------------------|
|      30 | 2020-02-08 12:08:29 |
|       8 | 2020-02-20 11:22:00 |
|       3 | 2020-02-20 18:00:00 |
|       4 | 2020-01-13 16:42:01 |
|       3 | 2020-02-20 11:22:01 |
|       4 | 2020-02-20 13:30:30 |
+---------+---------------------+
```

your query should return:

```
+---------+
| user_id |
+---------+
|       3 |
|       4 |
+---------+
```

User 3 logged in twice after the attack time. User 4 logged in at `2020-02-20 13:30:30`, which was after the attack time. Note that user 8 logged in exactly at the attack time, and thus will not receive the warning.

2. Given table:

```
+---------+---------------------+
| user_id |          login_time |
+---------+---------------------|
|       2 | 2020-01-01 00:00:00 |
|       2 | 2020-02-20 02:22:20 |
|       3 | 2020-02-20 11:22:01 |
|       4 | 2018-06-07 02:30:21 |
|       1 | 2021-05-20 08:24:11 |
|       1 | 2019-12-31 23:59:59 |
+---------+---------------------+
```

your query should return:

```
+---------+
| user_id |
+---------+
|       1 |
|       3 |
+---------+
```

Note that user 3 logged in exactly 1 second after the attack time.

3. Given table:

```
+---------+---------------------+
| user_id |          login_time |
+---------+---------------------|
|       6 | 2018-05-20 08:24:11 |
|       5 | 2019-12-31 23:59:59 |
|       4 | 2019-10-10 10:10:10 |
|       3 | 2020-02-19 02:22:20 |
|       2 | 2020-02-20 11:22:00 |
|       1 | 2020-02-20 11:21:59 |
+---------+---------------------+
```

Your query should return:

```
+---------+
| user_id |
+---------+
```

No one logged in after the attack time.

Assume that:

- column `user_id` contains only integers within the range [1..1,000];
- column `login_time` contains only timestamps between 2015-01-01 00:00:00 and 2025-12-31 23:59:59.

End of Challenge