DATABRICKS-CERTIFIED-ASSOCIAT

Q&As

Databricks Certified Associate Developer for Apache Spark 3.0

Pass Databricks DATABRICKS-CERTIFIED-ASSOCIATE-DEVELOPER-FOR-APACHE-SPARK Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.leads4pass.com/databricks-certified-associate-developer-for-apache-spark.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Databricks Official Exam Center

QUESTION 1

Which of the following is the idea behind dynamic partition pruning in Spark?

A. Dynamic partition pruning is intended to skip over the data you do not need in the results of a query.

B. Dynamic partition pruning concatenates columns of similar data types to optimize join performance.

C. Dynamic partition pruning performs wide transformations on disk instead of in memory.

D. Dynamic partition pruning reoptimizes physical plans based on data types and broadcast variables.

E. Dynamic partition pruning reoptimizes query plans based on runtime statistics collected during query execution.

Correct Answer: A

QUESTION 2

Which of the following describes a valid concern about partitioning?

A. A shuffle operation returns 200 partitions if not explicitly set.

B. Decreasing the number of partitions reduces the overall runtime of narrow transformations if there are more executors available than partitions.

C. No data is exchanged between executors when coalesce() is run.

D. Short partition processing times are indicative of low skew.

E. The coalesce() method should be used to increase the number of partitions.

Correct Answer: A

QUESTION 3

The code block shown below should return a new 2-column DataFrame that shows one attribute from column attributes per row next to the associated itemName, for all suppliers in column supplier whose name includes Sports. Choose the answer that correctly fills the blanks in the code block to accomplish this.

Sample of DataFrame itemsDf:

1.+------+----------------------------------+----------------------------+-------------------+

2.|itemId|itemName |attributes |supplier |

3.+------+----------------------------------+----------------------------+-------------------+

4.|1 |Thick Coat for Walking in the Snow|[blue, winter, cozy] |Sports Company Inc.| 5.|2 |Elegant Outdoors Summer Dress |[red, summer, fresh, cooling]|YetiX |

6.|3 |Outdoors Backpack |[green, summer, travel] |Sports Company Inc.|

7.+------+---------------------------------+----------------------------+-------------------+

Code block:

itemsDf.__1__(__2__).select(__3__, __4__) A. 1. filter

2.

col("supplier").isin("Sports")

3.

"itemName"

4.

explode(col("attributes"))

B. 1. where

2.

col("supplier").contains("Sports")

3.

"itemName"

4.

"attributes"

C. 1. where

2.

col(supplier).contains("Sports")

3.

explode(attributes)

4.

itemName

D. 1. where

2.

"Sports".isin(col("Supplier"))

3.

"itemName"

4.

array_explode("attributes")

E. 1. filter

2.

col("supplier").contains("Sports")

3.

"itemName"

4.

explode("attributes")

Correct Answer: E

Output of correct code block:

+----------+------+

|itemName |col |

+----------+------+

|Thick Coat for Walking in the Snow|blue |

|Thick Coat for Walking in the Snow|winter|

|Thick Coat for Walking in the Snow|cozy |

|Outdoors Backpack |green |

|Outdoors Backpack |summer|

|Outdoors Backpack |travel|

+----------+------+

The key to solving this is knowing about Spark\\'s explode operator. Using this operator, you can extract

values from arrays into single rows. The following guidance steps through

the

answers systematically from the first to the last gap. Note that there are many ways to solving the gap

QUESTION 4

Which of the following statements about reducing out-of-memory errors is incorrect?

A. Concatenating multiple string columns into a single column may guard against out-of- memory errors.

B. Reducing partition size can help against out-of-memory errors.

C. Limiting the amount of data being automatically broadcast in joins can help against out- of-memory errors.

D. Setting a limit on the maximum size of serialized data returned to the driver may help prevent out-ofmemory errors.

E. Decreasing the number of cores available to each executor can help against out-of- memory errors.

Correct Answer: A

Concatenating multiple string columns into a single column may guard against out-of- memory errors. Exactly, this is an incorrect answer! Concatenating any string columns does not reduce the size of the data, it just structures it a different way. This does little to how Spark processes the data and definitely does not reduce out-of-memory errors. Reducing partition size can help against out-of-memory errors. No, this is not incorrect. Reducing partition size is a viable way to aid against out-of- memory errors, since executors need to load partitions into memory before processing them. If the executor does not have enough memory available to do that, it will throw an out-of-memory error. Decreasing partition size can therefore be very helpful for preventing that. Decreasing the number of cores available to each executor can help against out-of- memory errors. No, this is not incorrect. To process a partition, this partition needs to be loaded into the memory of an executor. If you imagine that every core in every executor processes a partition, potentially in parallel with other executors, you can imagine that memory on the machine hosting the executors fills up quite quickly. So, memory usage of executors is a concern, especially when multiple partitions are processed at the same time. To strike a balance between performance and memory usage, decreasing the number of cores may help against out-of-memory errors. Setting a limit on the maximum size of serialized data returned to the driver may help prevent out-of-memory errors. No, this is not incorrect. When using commands like collect() that trigger the transmission of potentially large amounts of data from the cluster to the driver, the driver may experience out-of-memory errors. One strategy to avoid this is to be careful about using commands like collect() that send back large amounts of data to the driver. Another strategy is setting the parameter spark.driver.maxResultSize. If data to be transmitted to the driver exceeds the threshold specified by the parameter, Spark will abort the job and therefore prevent an out-of- memory error. Limiting the amount of data being automatically broadcast in joins can help against out-of- memory errors. Wrong, this is not incorrect. As part of Spark\\'s internal optimization, Spark may choose to speed up operations by broadcasting (usually relatively small) tables to executors. This broadcast is happening from the driver, so all the broadcast tables are loaded into the driver first. If these tables are relatively big, or multiple mid-size tables are being broadcast, this may lead to an out-ofmemory error. The maximum table size for which Spark will consider broadcasting is set by the spark.sql.autoBroadcastJoinThreshold parameter. More info: Configuration - Spark 3.1.2 Documentation and Spark OOM Error -- Closeup. Does the following look familiar when... | by Amit Singh Rathore | The Startup | Medium

QUESTION 5

Which of the following code blocks shuffles DataFrame transactionsDf, which has 8 partitions, so that it has 10 partitions?

A. transactionsDf.repartition(transactionsDf.getNumPartitions()+2)

B. transactionsDf.repartition(transactionsDf.rdd.getNumPartitions()+2)

C. transactionsDf.coalesce(10)

D. transactionsDf.coalesce(transactionsDf.getNumPartitions()+2)

E. transactionsDf.repartition(transactionsDf._partitions+2)

Correct Answer: B