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

The code block displayed below contains an error. The code block should trigger Spark to cache DataFrame transactionsDf in executor memory where available, writing to disk where insufficient

executor memory is available, in a fault-tolerant way. Find the error.

Code block:

transactionsDf.persist(StorageLevel.MEMORY_AND_DISK)

A. Caching is not supported in Spark, data are always recomputed.

B. Data caching capabilities can be accessed through the spark object, but not through the DataFrame API.

C. The storage level is inappropriate for fault-tolerant storage.

D. The code block uses the wrong operator for caching.

E. The DataFrameWriter needs to be invoked.

Correct Answer: C

The storage level is inappropriate for fault-tolerant storage. Correct. Typically, when thinking about fault tolerance and storage levels, you would want to store redundant copies of the dataset. This can be achieved by using a storage level such as StorageLevel.MEMORY_AND_DISK_2. The code block uses the wrong command for caching. Wrong. In this case, DataFrame.persist() needs to be used, since this operator supports passing a storage level. DataFrame.cache() does not support passing a storage level.

Caching is not supported in Spark, data are always recomputed. Incorrect. Caching is an important component of Spark, since it can help to accelerate Spark programs to great extent. Caching is often a good idea for datasets that need to be accessed repeatedly.

Data caching capabilities can be accessed through the spark object, but not through the DataFrame API. No. Caching is either accessed through DataFrame.cache() or DataFrame.persist().

The DataFrameWriter needs to be invoked. Wrong. The DataFrameWriter can be accessed via DataFrame.write and is used to write data to external data stores, mostly on disk. Here, we find keywords such as "cache" and "executor memory" that point us away from using external data stores. We aim to save data to memory to accelerate the reading process, since reading from disk is comparatively slower. The DataFrameWriter does not write to memory, so we cannot use it here.

More info: Best practices for caching in Spark SQL | by David Vrba | Towards Data Science

QUESTION 2

Which of the following statements about DAGs is correct?

A. DAGs help direct how Spark executors process tasks, but are a limitation to the proper execution of a query when an executor fails.

B. DAG stands for "Directing Acyclic Graph".

C. Spark strategically hides DAGs from developers, since the high degree of automation in Spark means that developers never need to consider DAG layouts.

D. In contrast to transformations, DAGs are never lazily executed.

E. DAGs can be decomposed into tasks that are executed in parallel.

Correct Answer: E

QUESTION 3

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 4

Which of the following DataFrame methods is classified as a transformation?

A. DataFrame.count()

B. DataFrame.show()

C. DataFrame.select()

D. DataFrame.foreach()

E. DataFrame.first()

Correct Answer: C

DataFrame.select()

Correct, DataFrame.select() is a transformation. When the command is executed, it is evaluated lazily and

returns an RDD when it is triggered by an action.

DataFrame.foreach()

Incorrect, DataFrame.foreach() is not a transformation, but an action. The intention of foreach() is to apply

code to each element of a DataFrame to update accumulator variables or write the

elements to external storage. The process does not return an RDD - it is an action! DataFrame.first()

Wrong. As an action, DataFrame.first() executed immediately and returns the first row of a DataFrame.

DataFrame.count()

Incorrect. DataFrame.count() is an action and returns the number of rows in a DataFrame.

DataFrame.show()

No, DataFrame.show() is an action and displays the DataFrame upon execution of the command.

QUESTION 5

The code block shown below should return all rows of DataFrame itemsDf that have at least 3 items in column itemNameElements. Choose the answer that correctly fills the blanks in the code block to accomplish this.

Example of DataFrame itemsDf:

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

2.|itemId|itemName |supplier |itemNameElements |

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

4.|1 |Thick Coat for Walking in the Snow|Sports Company Inc.|[Thick, Coat, for, Walking, in, the, Snow]|

5.|2 |Elegant Outdoors Summer Dress |YetiX |[Elegant, Outdoors, Summer, Dress] |

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

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

Code block:

itemsDf.__1__(__2__(__3__)__4__)

A. 1. select

2.

count

3.

col("itemNameElements")

4.

>3

B. 1. filter

2.

count

3.

itemNameElements

4.

>=3

C. 1. select

2.

count

3.

"itemNameElements"

4.

>3

D. 1. filter

2.

size

3.

"itemNameElements"

4.

>=3

E. 1. select

2.

size

3.

"itemNameElements"

4.

>3

Correct Answer: D

Correct code block:

itemsDf.filter(size("itemNameElements")>3)

Output of code block:

+------+---------------------------------+------------------+---------------------------------------+ |itemId|itemName |

supplier |itemNameElements |

+------+---------------------------------+------------------+---------------------------------------+ |1 |Thick Coat for

Walking in the Snow|Sports Company Inc.|[Thick, Coat, for, Walking, in, the, Snow]|

|2 |Elegant Outdoors Summer Dress |YetiX |[Elegant, Outdoors, Summer, Dress] |
+-----+---------------------------------+------------------+---------------------------------------+ The big difficulty with this is

in knowing the difference between count and size (refer to documentation below). size is the correct

function to choose here since it returns the number

of elements in an array on a per-row basis.

The other consideration for solving this is the difference between select and filter. Since we want to return

the rows in the original DataFrame, filter is the right choice. If we would use select, we would simply get a

single-column DataFrame showing which rows match the criteria, like so:

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

|(size(itemNameElements) > 3)|

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

|true |

|true |

|false |

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

More info:

Count documentation: pyspark.sql.functions.count -- PySpark 3.1.1 documentation Size documentation:

pyspark.sql.functions.size -- PySpark 3.1.1 documentation Static notebook | Dynamic notebook: See test

1, 47 (Databricks import instructions)

Latest DATABRICKS-CERTIFIED-ASSOCIATE-DEVELOPER-FOR-APACHE-SPARK Dumps

DATABRICKS-CERTIFIED-ASSOCIATE-DEVELOPER-FOR-APACHE-SPARK PDF Dumps

DATABRICKS-CERTIFIED-ASSOCIATE-DEVELOPER-FOR-APACHE-SPARK Practice Test