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.pass2lead.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 shown below should add column transactionDateForm to DataFrame transactionsDf. The column should express the unix-format timestamps in column transactionDate as string type like Apr 26 (Sunday). Choose the answer that correctly fills the blanks in the code block to accomplish this.

transactionsDf.__1__(__2__, from_unixtime(__3__, __4__))

A. 1. withColumn

2.

"transactionDateForm"

3.

"MMM d (EEEE)"

4.

"transactionDate"

B. 1. select

2.

"transactionDate"

3.

"transactionDateForm"

4.

"MMM d (EEEE)"

C. 1. withColumn

2.

"transactionDateForm"

3.

"transactionDate"

4.

"MMM d (EEEE)"

D. 1. withColumn

2.

"transactionDateForm"

3.

"transactionDate"

4.

"MM d (EEE)"

E. 1. withColumnRenamed

2.

"transactionDate"

3.

"transactionDateForm"

4.

"MM d (EEE)"

Correct Answer: C

QUESTION 2

Which of the following code blocks applies the Python function to_limit on column predError in table transactionsDf, returning a DataFrame with columns transactionId and result?

A. 1.spark.udf.register("LIMIT_FCN", to_limit) 2.spark.sql("SELECT transactionId, LIMIT_FCN(predError) AS result FROM transactionsDf")

B. 1.spark.udf.register("LIMIT_FCN", to_limit) 2.spark.sql("SELECT transactionId, LIMIT_FCN(predError) FROM transactionsDf AS result")

C. 1.spark.udf.register("LIMIT_FCN", to_limit) 2.spark.sql("SELECT transactionId, to_limit(predError) AS result FROM transactionsDf") spark.sql ("SELECT transactionId, udf(to_limit(predError)) AS result FROM transactionsDf")

D. 1.spark.udf.register(to_limit, "LIMIT_FCN") 2.spark.sql("SELECT transactionId, LIMIT_FCN(predError) AS result FROM transactionsDf")

Correct Answer: A

spark.udf.register("LIMIT_FCN", to_limit)

spark.sql("SELECT transactionId, LIMIT_FCN(predError) AS result FROM transactionsDf") Correct! First,

you have to register to_limit as UDF to use it in a sql statement. Then, you can use it under the

LIMIT_FCN name, correctly naming the resulting column result.

spark.udf.register(to_limit, "LIMIT_FCN")

spark.sql("SELECT transactionId, LIMIT_FCN(predError) AS result FROM transactionsDf") No. In this

answer, the arguments to spark.udf.register are flipped.

spark.udf.register("LIMIT_FCN", to_limit)

spark.sql("SELECT transactionId, to_limit(predError) AS result FROM transactionsDf") Wrong, this answer

does not use the registered LIMIT_FCN in the sql statement, but tries to access the to_limit method

directly. This will fail, since Spark cannot access it. spark.sql("SELECT transactionId, udf(to_limit

(predError)) AS result FROM transactionsDf") Incorrect, there is no udf method in Spark\\'s SQL.

spark.udf.register("LIMIT_FCN", to_limit)

spark.sql("SELECT transactionId, LIMIT_FCN(predError) FROM transactionsDf AS result") False. In this

answer, the column that results from applying the UDF is not correctly renamed to result.

Static notebook | Dynamic notebook: See test 3, 52 (Databricks import instructions)

QUESTION 3

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)

QUESTION 4

Which of the following code blocks writes DataFrame itemsDf to disk at storage location filePath, making sure to substitute any existing data at that location?

A. itemsDf.write.mode("overwrite").parquet(filePath)

B. itemsDf.write.option("parquet").mode("overwrite").path(filePath)

C. itemsDf.write(filePath, mode="overwrite")

D. itemsDf.write.mode("overwrite").path(filePath)

E. itemsDf.write().parquet(filePath, mode="overwrite")

Correct Answer: A

QUESTION 5

Which of the following statements about garbage collection in Spark is incorrect?

A. Garbage collection information can be accessed in the Spark UI\\'s stage detail view.

B. Optimizing garbage collection performance in Spark may limit caching ability.

C. Manually persisting RDDs in Spark prevents them from being garbage collected.

D. In Spark, using the G1 garbage collector is an alternative to using the default Parallel garbage collector.

E. Serialized caching is a strategy to increase the performance of garbage collection.

Correct Answer: C