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.passapply.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 code blocks can be used to save DataFrame transactionsDf to memory only, recalculating partitions that do not fit in memory when they are needed?

A. from pyspark import StorageLevel transactionsDf.cache(StorageLevel.MEMORY_ONLY)

B. transactionsDf.cache()

C. transactionsDf.storage_level(\\'MEMORY_ONLY\\')

D. transactionsDf.persist()

E. transactionsDf.clear_persist()

F. from pyspark import StorageLevel transactionsDf.persist(StorageLevel.MEMORY_ONLY)

Correct Answer: F

QUESTION 2

Which of the following code blocks shows the structure of a DataFrame in a tree-like way, containing both column names and types?

A. 1.print(itemsDf.columns) 2.print(itemsDf.types)

B. itemsDf.printSchema()

C. spark.schema(itemsDf)

D. itemsDf.rdd.printSchema()

E. itemsDf.print.schema()

Correct Answer: B

QUESTION 3

Which of the following code blocks returns a DataFrame where columns predError and productId are removed from DataFrame transactionsDf?

Sample of DataFrame transactionsDf:

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

2.|transactionId|predError|value|storeId|productId|f |

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

4.|1 |3 |4 |25 |1 |null|

5.|2 |6 |7 |2 |2 |null|

6.|3 |3 |null |25 |3 |null|

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

A. transactionsDf.withColumnRemoved("predError", "productId")

B. transactionsDf.drop(["predError", "productId", "associateId"])

C. transactionsDf.drop("predError", "productId", "associateId")

D. transactionsDf.dropColumns("predError", "productId", "associateId")

E. transactionsDf.drop(col("predError", "productId"))

Correct Answer: D

The key here is to understand that columns that are passed to DataFrame.drop() are ignored if they do not exist in the DataFrame. So, passing column name associateId to transactionsDf.drop() does not have any effect. Passing a list to transactionsDf.drop() is not valid. The documentation (link below) shows the call structure as DataFrame.drop(*cols). The * means that all arguments that are passed to DataFrame.drop() are read as columns. However, since a list of columns, for example ["predError", "productId", "associateId"] is not a column, Spark will run into an error. More info: pyspark.sql.DataFrame.drop -- PySpark 3.1.1 documentation Static notebook | Dynamic notebook: See test 1, 50 (Databricks import instructions)

QUESTION 4

Which of the following code blocks immediately removes the previously cached DataFrame transactionsDf from memory and disk?

A. array_remove(transactionsDf, "*")

B. transactionsDf.unpersist()

C. del transactionsDf

D. transactionsDf.clearCache()

E. transactionsDf.persist()

Correct Answer: B

transactionsDf.unpersist()

Correct. The DataFrame.unpersist() command does exactly what the asks for - it removes all cached

parts of the DataFrame from memory and disk.

del transactionsDf

False. While this option can help remove the DataFrame from memory and disk, it does not do so immediately. The reason is that this command just notifies the Python garbage collector that the transactionsDf now may be deleted from memory. However, the garbage collector does not do so immediately and, if you wanted it to run immediately, would need to be specifically triggered to do so. Find more information linked below.

array_remove(transactionsDf, "*")

Incorrect. The array_remove method from pyspark.sql.functions is used for removing elements from arrays in columns that match a specific condition. Also, the first argument would be a column, and not a DataFrame as shown in the code block.

transactionsDf.persist()

No. This code block does exactly the opposite of what is asked for: It caches (writes) DataFrame transactionsDf to memory and disk. Note that even though you do not pass in a specific storage level here, Spark will use the default storage level (MEMORY_AND_DISK).

transactionsDf.clearCache()

Wrong. Spark\\'s DataFrame does not have a clearCache() method.

More info: pyspark.sql.DataFrame.unpersist -- PySpark 3.1.2 documentation, python - How to delete an RDD in PySpark for the purpose of releasing resources? - Stack Overflow

Static notebook | Dynamic notebook: See test 3, 40 (Databricks import instructions)

QUESTION 5

Which of the following code blocks performs an inner join of DataFrames transactionsDf and itemsDf on columns productId and itemId, respectively, excluding columns value and storeId from DataFrame transactionsDf and column attributes from DataFrame itemsDf?

A. transactionsDf.drop(\\'value\\', \\'storeId\\').join(itemsDf.select(\\'attributes\\'), transactionsDf.productId==itemsDf.itemId)

B. 1.transactionsDf.createOrReplaceTempView(\\'transactionsDf\\') 2.itemsDf.createOrReplaceTempView(\\'itemsDf\\') 3.spark.sql("SELECT -value, -storeId FROM transactionsDf INNER JOIN itemsDf ON productId==itemId").drop("attributes")

C. transactionsDf.drop("value", "storeId").join(itemsDf.drop("attributes"), "transactionsDf.productId==itemsDf.itemId")

D. 1.transactionsDf \

2.

.drop(col(\\'value\\'), col(\\'storeId\\')) \

3.

.join(itemsDf.drop(col(\\'attributes\\')), col(\\'productId\\')==col(\\'itemId\\'))

E. 1.transactionsDf.createOrReplaceTempView(\\'transactionsDf\\') 2.itemsDf.createOrReplaceTempView(\\'itemsDf\\') 3.statement = """ 4.SELECT * FROM transactionsDf 5.INNER JOIN itemsDf 6.ON transactionsDf.productId==itemsDf.itemId

7."""

8.spark.sql(statement).drop("value", "storeId", "attributes")

Correct Answer: E