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.pass4itsure.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 returns a single-row DataFrame that only has a column corr which shows the Pearson correlation coefficient between columns predError and value in DataFrame transactionsDf?

A. transactionsDf.select(corr(["predError", "value"]).alias("corr")).first()

B. transactionsDf.select(corr(col("predError"), col("value")).alias("corr")).first()

C. transactionsDf.select(corr(predError, value).alias("corr"))

D. transactionsDf.select(corr(col("predError"), col("value")).alias("corr"))

E. transactionsDf.select(corr("predError", "value"))

Correct Answer: D

In difficulty, this is above what you can expect from the exam. What this wants to teach you, however, is

to pay attention to the useful details included in the

documentation.

pyspark.sql.corr is not a very common method, but it deals with Spark\\'s data structure in an interesting

way. The command takes two columns over multiple rows and returns a single row - similar to

an aggregation function. When examining the documentation (linked below), you will find this code

example:

a = range(20)

b = [2 * x for x in range(20)]

df = spark.createDataFrame(zip(a, b), ["a", "b"])

df.agg(corr("a", "b").alias(\\'c\\')).collect()

[Row(c=1.0)]

See how corr just returns a single row? Once you understand this, you should be suspicious about

answers that include first(), since there is no need to just select a single row. A reason to eliminate

those answers is that DataFrame.first() returns an object of type Row, but not DataFrame, as requested in

the question.

transactionsDf.select(corr(col("predError"), col("value")).alias("corr")) Correct! After calculating the Pearson

correlation coefficient, the resulting column is correctly renamed to corr.

transactionsDf.select(corr(predError, value).alias("corr")) No. In this answer, Python will interpret column

names predError and value as variable names.

transactionsDf.select(corr(col("predError"), col("value")).alias("corr")).first() Incorrect. first() returns a row,

not a DataFrame (see above and linked documentation below).

transactionsDf.select(corr("predError", "value"))

Wrong. Whie this statement returns a DataFrame in the desired shape, the column will have the name corr

(predError, value) and not corr.

transactionsDf.select(corr(["predError", "value"]).alias("corr")).first() False. In addition to first() returning a

row, this code block also uses the wrong call structure for command corr which takes two arguments (the

two columns to correlate).

More info:

-pyspark.sql.functions.corr -- PySpark 3.1.2 documentation

-pyspark.sql.DataFrame.first -- PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 3, 53 (Databricks import instructions)

QUESTION 2

Which of the following code blocks creates a new 6-column DataFrame by appending the rows of the 6column DataFrame yesterdayTransactionsDf to the rows of the 6-column DataFrame todayTransactionsDf, ignoring that both DataFrames have different column names?

A. union(todayTransactionsDf, yesterdayTransactionsDf)

B. todayTransactionsDf.unionByName(yesterdayTransactionsDf, allowMissingColumns=True)

C. todayTransactionsDf.unionByName(yesterdayTransactionsDf)

D. todayTransactionsDf.concat(yesterdayTransactionsDf)

E. todayTransactionsDf.union(yesterdayTransactionsDf)

Correct Answer: E

todayTransactionsDf.union(yesterdayTransactionsDf) Correct. The union command appends rows of yesterdayTransactionsDf to the rows of todayTransactionsDf, ignoring that both DataFrames have different column names. The resulting DataFrame will have the column names of DataFrame todayTransactionsDf. todayTransactionsDf.unionByName (yesterdayTransactionsDf) No. unionByName specifically tries to match columns in the two DataFrames by name and only appends values in columns with identical names across the two DataFrames. In the form presented above, the command is a great fit for joining DataFrames that have exactly the same columns, but in a different order. In this case though, the command will fail because the two DataFrames have different columns.

todayTransactionsDf.unionByName(yesterdayTransactionsDf, allowMissingColumns=True) No. The unionByName command is described in the previous explanation. However, with the allowMissingColumns argument set to True, it is no longer an issue that the two DataFrames have different column names. Any columns that do not have a match in the other DataFrame will be filled with null where there is no value. In the case at hand, the resulting DataFrame will have 7 or more columns though, so it this command is not the right answer. union(todayTransactionsDf, yesterdayTransactionsDf) No, there is no union method in pyspark.sql.functions. todayTransactionsDf.concat(yesterdayTransactionsDf) Wrong, the DataFrame class does not have a concat method. More info: pyspark.sql.DataFrame.union -- PySpark 3.1.2 documentation, pyspark.sql.DataFrame.unionByName -- PySpark 3.1.2 documentation Static notebook | Dynamic notebook: See test 3, 18 (Databricks import instructions)

QUESTION 3

The code block displayed below contains an error. The code block should return a new DataFrame that only contains rows from DataFrame transactionsDf in which the value in column predError is at least 5.

Find the error.

Code block:

transactionsDf.where("col(predError) >= 5")

A. The argument to the where method should be "predError >= 5".

B. Instead of where(), filter() should be used.

C. The expression returns the original DataFrame transactionsDf and not a new DataFrame. To avoid this, the code block should be transactionsDf.toNewDataFrame().where("col(predError) >= 5").

D. The argument to the where method cannot be a string.

E. Instead of >=, the SQL operator GEQ should be used.

Correct Answer: A

QUESTION 4

Which of the following code blocks returns a new DataFrame in which column attributes of DataFrame itemsDf is renamed to feature0 and column supplier to feature1?

A. itemsDf.withColumnRenamed(attributes, feature0).withColumnRenamed(supplier, feature1)

B. 1.itemsDf.withColumnRenamed("attributes", "feature0") 2.itemsDf.withColumnRenamed("supplier", "feature1")

C. itemsDf.withColumnRenamed(col("attributes"), col("feature0"), col("supplier"), col("feature1"))

D. itemsDf.withColumnRenamed("attributes", "feature0").withColumnRenamed("supplier", "feature1")

E. itemsDf.withColumn("attributes", "feature0").withColumn("supplier", "feature1")

Correct Answer: D

QUESTION 5

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