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

Which of the following code blocks returns about 150 randomly selected rows from the 1000-row DataFrame transactionsDf, assuming that any row can appear more than once in the returned DataFrame?

A. transactionsDf.resample(0.15, False, 3142)

B. transactionsDf.sample(0.15, False, 3142)

C. transactionsDf.sample(0.15)

D. transactionsDf.sample(0.85, 8429)

E. transactionsDf.sample(True, 0.15, 8261)

Correct Answer: E

Answering this correctly depends on whether you understand the arguments to the DataFrame.sample() method (link to the documentation below). The arguments are as follows: DataFrame.sample(withReplacement=None, fraction=None, seed=None). The first argument withReplacement specified whether a row can be drawn from the DataFrame multiple times. By default, this option is disabled in Spark. But we have to enable it here, since the question asks for a row being able to appear more than once. So, we need to pass True for this argument.

About replacement: "Replacement" is easiest explained with the example of removing random items from a box. When you remove those "with replacement" it means that after you have taken an item out of the box, you put it back inside. So, essentially, if you would randomly take 10 items out of a box with 100 items, there is a chance you take the same item twice or more times. "Without replacement" means that you would not put the item back into the box after removing it. So, every time you remove an item from the box, there is one less item in the box and you can never take the same item twice. The second argument to the withReplacement method is fraction. This referes to the fraction of items that should be returned. In the we are asked for 150 out of 1000 items ?a fraction of 0.15. The last argument is a random seed. A random seed makes a randomized processed repeatable. This means that if you would re-run the same sample() operation with the same random seed, you would get the same rows returned from the sample() command. There is no behavior around the random seed specified in the question. The varying random seeds are only there to confuse you!

More info: pyspark.sql.DataFrame.sample -- PySpark 3.1.1 documentation Static notebook | Dynamic notebook: See test 1, 49 (Databricks import instructions)

QUESTION 2

Which of the following code blocks returns only rows from DataFrame transactionsDf in which values in column productId are unique?

A. transactionsDf.distinct("productId")

B. transactionsDf.dropDuplicates(subset=["productId"])

C. transactionsDf.drop_duplicates(subset="productId")

D. transactionsDf.unique("productId")

E. transactionsDf.dropDuplicates(subset="productId")

Correct Answer: B

Although the suggests using a method called unique() here, that method does not actually exist in PySpark. In PySpark, it is called distinct(). But then, this method is not the right one to use here, since with distinct() we could filter out unique values in a specific column. However, we want to return the entire rows here. So the trick is to use dropDuplicates with the subset keyword parameter. In the documentation for dropDuplicates, the examples show that subset should be used with a list. And this is exactly the key to solving this question: The productId column needs to be fed into the subset argument in a list, even though it is just a single column. More info: pyspark.sql.DataFrame.dropDuplicates -- PySpark 3.1.1 documentation Static notebook | Dynamic notebook: See test 1, 45 (Databricks import instructions)

QUESTION 3

Which of the following code blocks prints out in how many rows the expression Inc. appears in the stringtype column supplier of DataFrame itemsDf?

A. 1.counter = 0

2.

3.for index, row in itemsDf.iterrows():

4.

if \\'Inc.\\' in row[\\'supplier\\']:

5.

counter = counter + 1

6.

7.print(counter)

B. 1.counter = 0

2.

3.def count(x):

4.

if \\'Inc.\\' in x[\\'supplier\\']:

5.

counter = counter + 1

6.

7.itemsDf.foreach(count)

8.print(counter)

C. print(itemsDf.foreach(lambda x: \\'Inc.\\' in x))

D. print(itemsDf.foreach(lambda x: \\'Inc.\\' in x).sum())

E. 1.accum=sc.accumulator(0)

2.

3.def check_if_inc_in_supplier(row):

4.

if \\'Inc.\\' in row[\\'supplier\\']:

5.

accum.add(1)

6.

7.itemsDf.foreach(check_if_inc_in_supplier)

8.print(accum.value)

Correct Answer: E

Correct code block:

accum=sc.accumulator(0)

def check_if_inc_in_supplier(row):

if \\'Inc.\\' in row[\\'supplier\\']:

accum.add(1)

itemsDf.foreach(check_if_inc_in_supplier)

print(accum.value)

To answer this correctly, you need to know both about the DataFrame.foreach() method and

accumulators.

When Spark runs the code, it executes it on the executors. The executors do not have any information

about variables outside of their scope. This is whhy simply using a Python variable counter,

like in the two examples that start with counter = 0, will not work. You need to tell the executors explicitly

that counter is a special shared variable, an Accumulator, which is managed by the driver

and can be accessed by all executors for the purpose of adding to it. If you have used Pandas in the past,

you might be familiar with the iterrows() command.

Notice that there is no such command in PySpark.

The two examples that start with print do not work, since DataFrame.foreach() does not have a return

value.

More info: pyspark.sql.DataFrame.foreach -- PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 3, 22 (Databricks import instructions)

QUESTION 4

The code block displayed below contains an error. The code block is intended to join DataFrame itemsDf with the larger DataFrame transactionsDf on column itemId. Find the error.

Code block:

transactionsDf.join(itemsDf, "itemId", how="broadcast")

A. The syntax is wrong, how= should be removed from the code block.

B. The join method should be replaced by the broadcast method.

C. Spark will only perform the broadcast operation if this behavior has been enabled on the Spark cluster.

D. The larger DataFrame transactionsDf is being broadcasted, rather than the smaller DataFrame itemsDf.

E. broadcast is not a valid join type.

Correct Answer: E

QUESTION 5

The code block shown below should return a copy of DataFrame transactionsDf with an added column

cos. This column should have the values in column value converted to degrees and having the cosine of

those converted values taken, rounded to two decimals. Choose the answer that correctly fills the blanks in

the code block to accomplish this.

Code block:

transactionsDf.__1__(__2__, round(__3__(__4__(__5__)),2))

A. 1. withColumn

2.

col("cos")

3.

cos

4.

degrees

5.

transactionsDf.value

B. 1. withColumnRenamed

2.

"cos"

3.

cos

4.

degrees

5.

"transactionsDf.value"

C. 1. withColumn

2.

"cos"

3.

cos

4.

degrees

5.

transactionsDf.value

D. 1. withColumn

2.

col("cos")

3.

cos

4.

degrees

5.

col("value")

E. 1. withColumn

2.

"cos"

3.

degrees

4.

cos

5.

col("value")

Correct Answer: C