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

QUESTION 1

Which of the following code blocks returns all unique values of column storeId in DataFrame transactionsDf?

A. transactionsDf["storeId"].distinct()

B. transactionsDf.select("storeId").distinct()

C. transactionsDf.filter("storeId").distinct()

D. transactionsDf.select(col("storeId").distinct())

E. transactionsDf.distinct("storeId")

Correct Answer: B

distinct() is a method of a DataFrame. Knowing this, or recognizing this from the documentation, is the key to solving this question. More info: pyspark.sql.DataFrame.distinct -- PySpark 3.1.2 documentation Static notebook | Dynamic notebook: See test 2, 19 (Databricks import instructions)

QUESTION 2

Which of the following describes Spark\\'s way of managing memory?

A. Spark uses a subset of the reserved system memory.

B. Storage memory is used for caching partitions derived from DataFrames.

C. As a general rule for garbage collection, Spark performs better on many small objects than few big objects.

D. Disabling serialization potentially greatly reduces the memory footprint of a Spark application.

E. Spark\\'s memory usage can be divided into three categories: Execution, transaction, and storage.

Correct Answer: B

Spark\\'s memory usage can be divided into three categories: Execution, transaction, and storage.

No, it is either execution or storage.

As a general rule for garbage collection, Spark performs better on many small objects than few big objects.

No, Spark\\'s garbage collection runs faster on fewer big objects than many small objects. Disabling

serialization potentially greatly reduces the memory footprint of a Spark application.

The opposite is true ?serialization reduces the memory footprint, but may impact performance in a

negative way.

Spark uses a subset of the reserved system memory. No, the reserved system memory is separate from

Spark memory. Reserved memory stores Spark\\'s internal objects.

More info: Tuning - Spark 3.1.2 Documentation, Spark Memory Management | Distributed Systems

Architecture, Learning Spark, 2nd Edition, Chapter 7

QUESTION 3

Which of the following is not a feature of Adaptive Query Execution?

A. Replace a sort merge join with a broadcast join, where appropriate.

B. Coalesce partitions to accelerate data processing.

C. Split skewed partitions into smaller partitions to avoid differences in partition processing time.

D. Reroute a query in case of an executor failure.

E. Collect runtime statistics during query execution.

Correct Answer: D

QUESTION 4

Which of the following code blocks returns a single row from DataFrame transactionsDf?

Full 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.| 4| null| null| 3| 2|null|
8.| 5| null| null| null| 2|null|
9.| 6| 3| 2| 25| 2|null|
10.+-------------+---------+-----+-------+---------+----+
```

A. transactionsDf.where(col("storeId").between(3,25))

B. transactionsDf.filter((col("storeId")!=25) | (col("productId")==2))

C. transactionsDf.filter(col("storeId")==25).select("predError","storeId").distinct()

D. transactionsDf.select("productId", "storeId").where("storeId == 2 OR storeId != 25")

E. transactionsDf.where(col("value").isNull()).select("productId", "storeId").distinct()

Correct Answer: C

QUESTION 5

Which of the following code blocks uses a schema fileSchema to read a parquet file at location filePath into a DataFrame?

A. spark.read.schema(fileSchema).format("parquet").load(filePath)

B. spark.read.schema("fileSchema").format("parquet").load(filePath)

C. spark.read().schema(fileSchema).parquet(filePath)

D. spark.read().schema(fileSchema).format(parquet).load(filePath)

E. spark.read.schema(fileSchema).open(filePath)

Correct Answer: A

Pay attention here to which variables are quoted. fileSchema is a variable and thus should not be in quotes. parquet is not a variable and therefore should be in quotes. SparkSession.read (here referenced as spark.read) returns a DataFrameReader which all subsequent calls reference - the DataFrameReader is not callable, so you should not use parentheses here. Finally, there is no open method in PySpark. The method name is load. Static notebook | Dynamic notebook: See test 1, 44 (Databricks import instructions)

QUESTION 6

Which of the following describes the role of tasks in the Spark execution hierarchy?

A. Tasks are the smallest element in the execution hierarchy.

B. Within one task, the slots are the unit of work done for each partition of the data.

C. Tasks are the second-smallest element in the execution hierarchy.

D. Stages with narrow dependencies can be grouped into one task.

E. Tasks with wide dependencies can be grouped into one stage.

Correct Answer: A

Stages with narrow dependencies can be grouped into one task. Wrong, tasks with narrow dependencies can be grouped into one stage. Tasks with wide dependencies can be grouped into one stage. Wrong, since a wide transformation causes a shuffle which always marks the boundary of a stage. So, you cannot bundle multiple tasks that have wide dependencies into a stage. Tasks are the second-smallest element in the execution hierarchy. No, they are the smallest element in the execution hierarchy. Within one task, the slots are the unit of work done for each partition of the data. No, tasks are the unit of work done per partition. Slots help Spark parallelize work. An executor can have multiple slots which enable it to process multiple tasks in parallel.

QUESTION 7

Which of the following code blocks creates a new one-column, two-row DataFrame dfDates with column date of type timestamp?

A. 1.dfDates = spark.createDataFrame(["23/01/2022 11:28:12","24/01/2022 10:58:34"], ["date"]) 2.dfDates = dfDates.withColumn("date", to_timestamp("dd/MM/yyyy HH:mm:ss", "date"))

B. 1.dfDates = spark.createDataFrame([("23/01/2022 11:28:12",),("24/01/2022 10:58:34",)], ["date"]) 2.dfDates = dfDates.withColumnRenamed("date", to_timestamp("date", "yyyy-MM-ddHH:mm:ss"))

C. 1.dfDates = spark.createDataFrame([("23/01/2022 11:28:12",),("24/01/2022 10:58:34",)], ["date"]) 2.dfDates = dfDates.withColumn("date", to_timestamp("date", "dd/MM/yyyy HH:mm:ss"))

D. 1.dfDates = spark.createDataFrame(["23/01/2022 11:28:12","24/01/2022 10:58:34"], ["date"]) 2.dfDates = dfDates.withColumnRenamed("date", to_datetime("date", "yyyy-MM-ddHH:mm:ss"))

E. 1.dfDates = spark.createDataFrame([("23/01/2022 11:28:12",),("24/01/2022 10:58:34",)], ["date"])

Correct Answer: C

QUESTION 8

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 9

Which of the following code blocks returns the number of unique values in column storeId of DataFrame transactionsDf?

A. transactionsDf.select("storeId").dropDuplicates().count()

B. transactionsDf.select(count("storeId")).dropDuplicates()

C. transactionsDf.select(distinct("storeId")).count()

D. transactionsDf.dropDuplicates().agg(count("storeId"))

E. transactionsDf.distinct().select("storeId").count()

Correct Answer: A

transactionsDf.select("storeId").dropDuplicates().count() Correct! After dropping all duplicates from column storeId, the remaining rows get counted, representing the number of unique values in the column. transactionsDf.select(count("storeId")).dropDuplicates() No. transactionsDf.select(count("storeId")) just returns a single-row DataFrame showing the number of non-null rows. dropDuplicates() does not have any effect in this context.

transactionsDf.dropDuplicates().agg(count("storeId")) Incorrect. While transactionsDf.dropDuplicates() removes duplicate rows from transactionsDf, it does not do so taking only column storeId into consideration, but eliminates full row duplicates instead. transactionsDf.distinct().select("storeId").count() Wrong. transactionsDf.distinct() identifies unique rows across all columns, but not only unique rows with respect to column storeId. This may leave duplicate values in the column, making the count not represent the number of unique values in that column. transactionsDf.select(distinct("storeId")).count() False. There is no distinct method in pyspark.sql.functions.

QUESTION 10

Which is the highest level in Spark\\'s execution hierarchy?

A. Task

B. Executor

C. Slot

D. Job

E. Stage

Correct Answer: D

QUESTION 11

Which of the following code blocks silently writes DataFrame itemsDf in avro format to location fileLocation if a file does not yet exist at that location?

A. itemsDf.write.avro(fileLocation)

B. itemsDf.write.format("avro").mode("ignore").save(fileLocation)

C. itemsDf.write.format("avro").mode("errorifexists").save(fileLocation)

D. itemsDf.save.format("avro").mode("ignore").write(fileLocation)

E. spark.DataFrameWriter(itemsDf).format("avro").write(fileLocation)

Correct Answer: A

QUESTION 12

The code block shown below should return the number of columns in the CSV file stored at location filePath. From the CSV file, only lines should be read that do not start with a # character. Choose the answer that correctly fills the blanks in the code block to accomplish this.

Code block:

```
__1__(__2__.__3__.csv(filePath, __4__).__5__)
```

A. 1. size

2.

spark

3.

read()

4.

escape=\\'#\\'

5.

columns

B. 1. DataFrame

2.

spark

3.

read()

4.

escape=\\'#\\'

5.

shape[0]

C. 1. len

2.

pyspark

3.

DataFrameReader

4.

comment=\\'#\\'

5.

columns

D. 1. size

2.

pyspark

3.

DataFrameReader

4.

comment=\\'#\\'

5.

columns

E. 1. len

2.

spark

3.

read

4.

comment=\\'#\\'

5.

columns

Correct Answer: E

QUESTION 13

The code block shown below should return a DataFrame with all columns of DataFrame transactionsDf, but only maximum 2 rows in which column productId has at least the value 2. Choose the answer that correctly fills the blanks in the code block to accomplish this.

transactionsDf.__1__(__2__).__3__

A. 1. where

2.

"productId" > 2

3.

max(2)

B. 1. where

2.

transactionsDf[productId] >= 2

3.

limit(2)

C. 1. filter

2.

productId > 2

3.

max(2)

D. 1. filter

2.

col("productId") >= 2

3.

limit(2)

E. 1. where

2.

productId >= 2

3.

limit(2)

Correct Answer: D

Correct code block:

transactionsDf.filter(col("productId") >= 2).limit(2) The filter and where operators in gap 1 are just aliases of

one another, so you cannot use them to pick the right answer.

The column definition in gap 2 is more helpful. The DataFrame.filter() method takes an argument of type Column or str. From all possible answers, only the one including col("productId") >= 2 fits this profile, since it returns a Column type. The answer option using "productId" > 2 is invalid, since Spark does not understand that "productId" refers to column productId. The answer option using transactionsDf[productId] >= 2 is wrong because you cannot refer to a column using square bracket notation in Spark (if you are coming from Python using Pandas, this is something to watch out for). In all other options, productId is being referred to as a Python variable, so they are relatively easy to eliminate. Also note that the asks for the value in column productId being at least 2. This translates to a "greater or equal" sign (>= 2), but not a "greater" sign (> 2). Another thing worth noting is that there is no DataFrame.max() method. If you picked any option including this, you may be confusing it with the pyspark.sql.functions.max method. The correct method to limit the amount of rows is the DataFrame.limit() method.

More info:

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

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

Static notebook | Dynamic notebook: See test 3, 54 (Databricks import instructions)

QUESTION 14

Which of the following code blocks returns a single-column DataFrame of all entries in Python list throughputRates which contains only float-type values ?

A. spark.createDataFrame((throughputRates), FloatType)

B. spark.createDataFrame(throughputRates, FloatType)

C. spark.DataFrame(throughputRates, FloatType)

D. spark.createDataFrame(throughputRates)

E. spark.createDataFrame(throughputRates, FloatType())

Correct Answer: E

spark.createDataFrame(throughputRates, FloatType()) Correct! spark.createDataFrame is the correct

operator to use here and the type FloatType() which is passed in for the command\\'s schema argument is

correctly instantiated using the parentheses.

Remember that it is essential in PySpark to instantiate types when passing them to

SparkSession.createDataFrame. And, in Databricks, spark returns a SparkSession object.

spark.createDataFrame((throughputRates), FloatType) No. While packing throughputRates in parentheses

does not do anything to the execution of this command, not instantiating the FloatType with parentheses

as in the previous answer will make this

command fail.

spark.createDataFrame(throughputRates, FloatType)

Incorrect. Given that it does not matter whether you pass throughputRates in parentheses or not, see the

explanation of the previous answer for further insights.

spark.DataFrame(throughputRates, FloatType)

Wrong. There is no SparkSession.DataFrame() method in Spark.

spark.createDataFrame(throughputRates)

False. Avoiding the schema argument will have PySpark try to infer the schema. However, as you can see

in the documentation (linked below), the inference will only work if you pass in an "RDD of

either Row, namedtuple, or dict" for data (the first argument to createDataFrame). But since you are

passing a Python list, Spark\\'s schema inference will fail.

More info: pyspark.sql.SparkSession.createDataFrame -- PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 3, 55 (Databricks import instructions)

QUESTION 15

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)