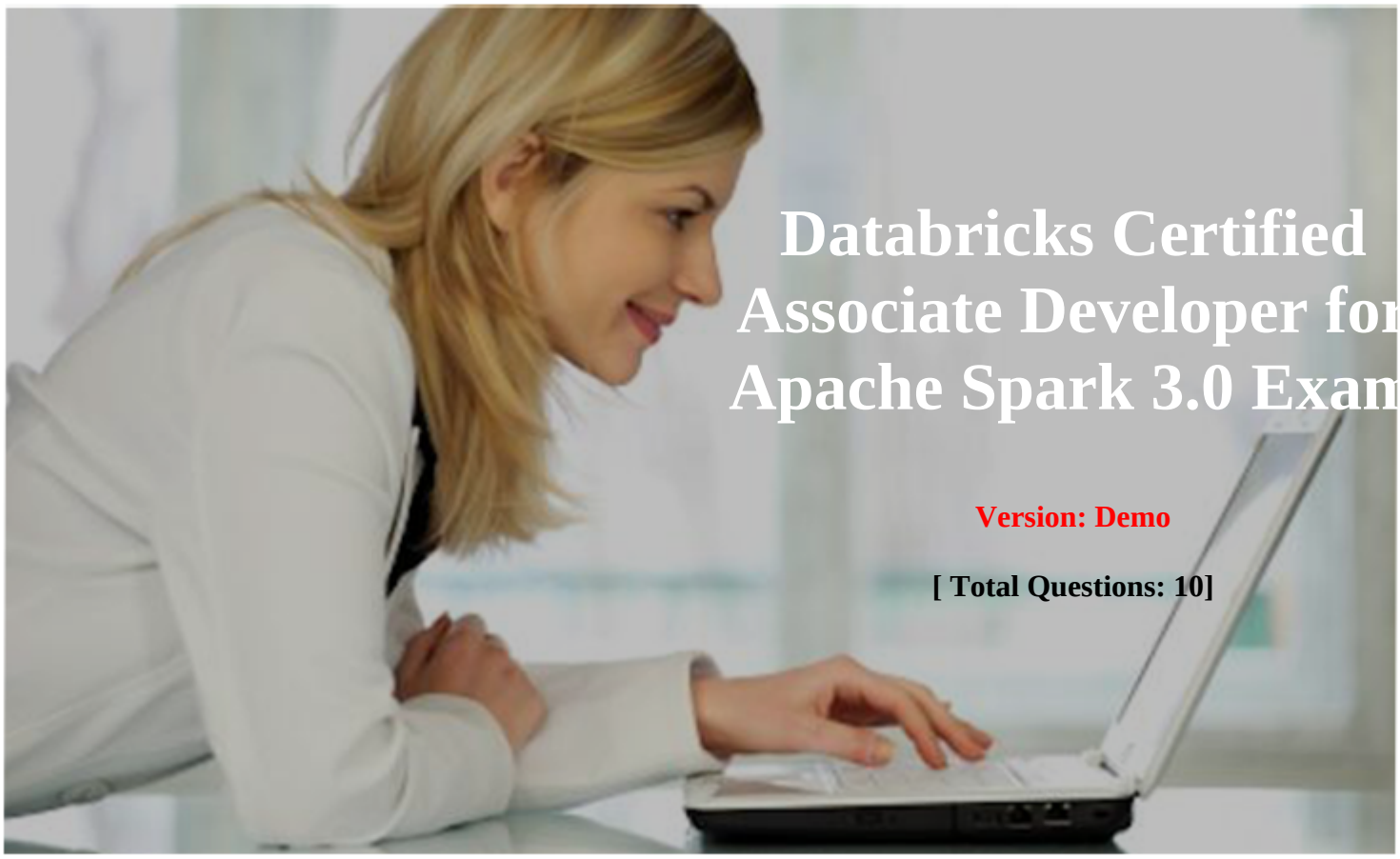
DumpsCafe

Databricks

abricks-Certified-Associate-Developer-for-Apache-Spark



Web: www.dumpscafe.com

Email: support@dumpscafe.com

IMPORTANT NOTICE

Feedback

We have developed quality product and state-of-art service to ensure our customers interest. If you have any suggestions, please feel free to contact us at feedback@dumpscafe.com

Support

If you have any questions about our product, please provide the following items:

- exam code
- screenshot of the question
- login id/email

please contact us at support@dumpscafe.com and our technical experts will provide support within 24 hours.

Copyright

Question #:1

The code block displayed below contains an error. The code block should return a copy of DataFrame transactionsDf where the name of column transactionId has been changed to

transactionNumber. Find the error.

Code block:

transactionsDf.withColumn("transactionNumber", "transactionId")

A. The arguments to the withColumn method need to be reordered.

B. The arguments to the withColumn method need to be reordered and the copy() operator should be appended to the code block to ensure a copy is returned.

C. The copy() operator should be appended to the code block to ensure a copy is returned.

D. Each column name needs to be wrapped in the col() method and method withColumn should be replaced by method withColumnRenamed.

E. The method withColumn should be replaced by method withColumnRenamed and the arguments to the method need to be reordered.

Answer: E

Explanation

Explanation

Correct code block:

transactionsDf.withColumnRenamed("transactionId", "transactionNumber")

Note that in Spark, a copy is returned by default. So, there is no need to append copy() to the code block.

More info: pyspark.sql.DataFrame.withColumnRenamed — PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 2, QUESTION NO: 26 (Databricks import instructions)

Question #:2

Which of the following code blocks returns a DataFrame that is an inner join of DataFrame itemsDf and DataFrame transactionsDf, on columns itemId and productId, respectively and in which every

itemId just appears once?

A. itemsDf.join(transactionsDf, "itemsDf.itemId==transactionsDf.productId").distinct("itemId")

B. itemsDf.join(transactionsDf, itemsDf.itemId==transactionsDf.productId).dropDuplicates(["itemId"])

C. itemsDf.join(transactionsDf, itemsDf.itemId==transactionsDf.productId).dropDuplicates("itemId")

D. itemsDf.join(transactionsDf, itemsDf.itemId==transactionsDf.productId, how="inner").distinct(["itemId"])

E. itemsDf.join(transactionsDf, "itemsDf.itemId==transactionsDf.productId", how="inner").dropDuplicates(["itemId"])

Answer: B

Explanation

Explanation

Filtering out distinct rows based on columns is achieved with the dropDuplicates method, not the distinct method which does not take any arguments.

The second argument of the join() method only accepts strings if they are column names. The SQL-like statement "itemsDf.itemId==transactionsDf.productId" is therefore invalid.

In addition, it is not necessary to specify how="inner", since the default join type for the join command is already inner.

More info: pyspark.sql.DataFrame.join — PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 2, QUESTION NO: 53 (Databricks import instructions)

Question #:3

The code block shown below should return a DataFrame with only columns from DataFrame transactionsDf for which there is a corresponding transactionId in DataFrame itemsDf. DataFrame

itemsDf is very small and much smaller than DataFrame transactionsDf. The query should be executed in an optimized way. Choose the answer that correctly fills the blanks in the code block to

accomplish this.

__1__.__2__(__3__, __4__, __5__)

A. 1. transactionsDf

2. join

3. broadcast(itemsDf)

4. transactionsDf.transactionId==itemsDf.transactionId

5. "outer"

B. 1. transactionsDf
2. join
3. itemsDf
4. transactionsDf.transactionId==itemsDf.transactionId
5. "anti"

C. 1. transactionsDf
2. join
3. broadcast(itemsDf)
4. "transactionId"
5. "left_semi"

D. 1. itemsDf
2. broadcast
3. transactionsDf
4. "transactionId"
5. "left_semi"

E. 1. itemsDf
2. join
3. broadcast(transactionsDf)
4. "transactionId"
5. "left_semi"

Answer: C

Explanation

Explanation

Correct code block:

transactionsDf.join(broadcast(itemsDf), "transactionId", "left_semi")

This QUESTION NO: is extremely difficult and exceeds the difficulty of questions in the exam by far.

A first indication of what is asked from you here is the remark that "the query should be executed in an optimized way". You also have qualitative information about the size of itemsDf and

transactionsDf. Given that itemsDf is "very small" and that the execution should be optimized, you should consider instructing Spark to perform a broadcast join, broadcasting the "very small"

DataFrame itemsDf to all executors. You can explicitly suggest this to Spark via wrapping itemsDf into a broadcast() operator. One answer option does not include this operator, so you can disregard

it. Another answer option wraps the broadcast() operator around transactionsDf - the bigger of the two DataFrames. This answer option does not make sense in the optimization context and can

likewise be disregarded.

When thinking about the broadcast() operator, you may also remember that it is a method of pyspark.sql.functions. One answer option, however, resolves to itemsDf.broadcast([...]). The DataFrame

class has no broadcast() method, so this answer option can be eliminated as well.

All two remaining answer options resolve to transactionsDf.join([...]) in the first 2 gaps, so you will have to figure out the details of the join now. You can pick between an outer and a left semi join. An

outer join would include columns from both DataFrames, where a left semi join only includes columns from the "left" table, here transactionsDf, just as asked for by the question. So, the correct

answer is the one that uses the left_semi join.

Question #:4

Which of the following code blocks creates a new DataFrame with two columns season and wind_speed_ms where column season is of data type string and column wind_speed_ms is of data type

double?

A. spark.DataFrame({"season": ["winter","summer"], "wind_speed_ms": [4.5, 7.5]})

B. spark.createDataFrame([("summer", 4.5), ("winter", 7.5)], ["season", "wind_speed_ms"])

C. 1. from pyspark.sql import types as T

2. spark.createDataFrame((("summer", 4.5), ("winter", 7.5)), T.StructType([T.StructField("season", T.CharType()), T.StructField("season", T.DoubleType())]))

D. spark.newDataFrame([("summer", 4.5), ("winter", 7.5)], ["season", "wind_speed_ms"])

E. spark.createDataFrame({"season": ["winter","summer"], "wind_speed_ms": [4.5, 7.5]})

Answer: B

Explanation

Explanation

spark.createDataFrame([("summer", 4.5), ("winter", 7.5)], ["season", "wind_speed_ms"])

Correct. This command uses the Spark Session's createDataFrame method to create a new DataFrame. Notice how rows, columns, and column names are passed in here: The rows are specified

as a Python list. Every entry in the list is a new row. Columns are specified as Python tuples (for example ("summer", 4.5)). Every column is one entry in the tuple.

The column names are specified as the second argument to createDataFrame(). The documentation (link below) shows that "when schema is a list of column names, the type of each column will be

inferred from data" (the first argument). Since values 4.5 and 7.5 are both float variables, Spark will correctly infer the double type for column wind_speed_ms. Given that all values in column

"season" contain only strings, Spark will cast the column appropriately as string.

Find out more about SparkSession.createDataFrame() via the link below.

spark.newDataFrame([("summer", 4.5), ("winter", 7.5)], ["season", "wind_speed_ms"])

No, the SparkSession does not have a newDataFrame method.

from pyspark.sql import types as T

spark.createDataFrame((("summer", 4.5), ("winter", 7.5)), T.StructType([T.StructField("season", T.CharType()), T.StructField("season", T.DoubleType())]))

No. pyspark.sql.types does not have a CharType type. See link below for available data types in Spark.

spark.createDataFrame({"season": ["winter","summer"], "wind_speed_ms": [4.5, 7.5]})

No, this is not correct Spark syntax. If you have considered this option to be correct, you may have some experience with Python's pandas package, in which this would be correct syntax. To create

a Spark DataFrame from a Pandas DataFrame, you can simply use spark.createDataFrame(pandasDf) where pandasDf is the Pandas DataFrame.

Find out more about Spark syntax options using the examples in the documentation for SparkSession.createDataFrame linked below.

spark.DataFrame({"season": ["winter","summer"], "wind_speed_ms": [4.5, 7.5]})

No, the Spark Session (indicated by spark in the code above) does not have a DataFrame method.

More info: pyspark.sql.SparkSession.createDataFrame — PySpark 3.1.1 documentation and Data Types - Spark 3.1.2 Documentation

Static notebook | Dynamic notebook: See test 1, QUESTION NO: 41 (Databricks import instructions)

Question #:5

The code block shown below should return a DataFrame with two columns, itemId and col. In this DataFrame, for each element in column attributes of DataFrame itemDf there should be a separate

row in which the column itemId contains the associated itemId from DataFrame itemsDf. The new DataFrame should only contain rows for rows in DataFrame itemsDf in which the column attributes

contains the element cozy.

A sample of DataFrame itemsDf is below.

Code block:

itemsDf.__1__(__2__).__3__(__4__, __5__(__6__))

A. 1. filter
   2. array_contains("cozy")
   3. select
   4. "itemId"
   5. explode
   6. "attributes"

B. 1. where
   2. "array_contains(attributes, 'cozy')"
   3. select
   4. itemId
   5. explode
   6. attributes

C. 1. filter
2. "array_contains(attributes, 'cozy')"
3. select
4. "itemId"
5. map
6. "attributes"

D. 1. filter
2. "array_contains(attributes, cozy)"
3. select
4. "itemId"
5. explode
6. "attributes"

E. 1. filter
2. "array_contains(attributes, 'cozy')"
3. select
4. "itemId"
5. explode
6. "attributes"

Answer: E

Explanation

The correct code block is:

itemsDf.filter("array_contains(attributes, 'cozy')").select("itemId", explode("attributes"))

The key here is understanding how to use array_contains(). You can either use it as an expression in a string, or you can import it from pyspark.sql.functions. In that case, the following would also

work:

itemsDf.filter(array_contains("attributes", "cozy")).select("itemId", explode("attributes"))

Static notebook | Dynamic notebook: See test 1, QUESTION NO: 29 (Databricks import instructions) (https://flrs.github.io/spark_practice_tests_code/#1/29.html ,

https://bit.ly/sparkpracticeexams_import_instructions)

Question #:6

Which of the following is a viable way to improve Spark's performance when dealing with large amounts of data, given that there is only a single application running on the cluster?

A. Increase values for the properties spark.default.parallelism and spark.sql.shuffle.partitions

B. Decrease values for the properties spark.default.parallelism and spark.sql.partitions

C. Increase values for the properties spark.sql.parallelism and spark.sql.partitions

D. Increase values for the properties spark.sql.parallelism and spark.sql.shuffle.partitions

E. Increase values for the properties spark.dynamicAllocation.maxExecutors, spark.default.parallelism, and spark.sql.shuffle.partitions

Answer: A

Explanation

Decrease values for the properties spark.default.parallelism and spark.sql.partitions

No, these values need to be increased.

Increase values for the properties spark.sql.parallelism and spark.sql.partitions

Wrong, there is no property spark.sql.parallelism.

Increase values for the properties spark.sql.parallelism and spark.sql.shuffle.partitions

See above.

Increase values for the properties spark.dynamicAllocation.maxExecutors, spark.default.parallelism, and spark.sql.shuffle.partitions

The property spark.dynamicAllocation.maxExecutors is only in effect if dynamic allocation is enabled, using the spark.dynamicAllocation.enabled property. It is disabled by default. Dynamic

allocation can be useful when to run multiple applications on the same cluster in parallel. However, in this case there is only a single application running on the cluster, so enabling dynamic

allocation would not yield a performance benefit.

More info: Practical Spark Tips For Data Scientists | Experfy.com and Basics of Apache Spark Configuration Settings | by Halil Ertan | Towards Data Science (https://bit.ly/3gA0A6w ,

https://bit.ly/2QxhNTr)

Question #:7

Which of the following describes a shuffle?

A. A shuffle is a process that is executed during a broadcast hash join.

B. A shuffle is a process that compares data across executors.

C. A shuffle is a process that compares data across partitions.

D. A shuffle is a Spark operation that results from DataFrame.coalesce().

E. A shuffle is a process that allocates partitions to executors.

Answer: C

Explanation

A shuffle is a Spark operation that results from DataFrame.coalesce().

No. DataFrame.coalesce() does not result in a shuffle.

A shuffle is a process that allocates partitions to executors.

This is incorrect.

A shuffle is a process that is executed during a broadcast hash join.

No, broadcast hash joins avoid shuffles and yield performance benefits if at least one of the two tables is small in size (<= 10 MB by default). Broadcast hash joins can avoid shuffles because

instead of exchanging partitions between executors, they broadcast a small table to all executors that then perform the rest of the join operation locally.

A shuffle is a process that compares data across executors.

No, in a shuffle, data is compared across partitions, and not executors.

More info: Spark Repartition & Coalesce - Explained (https://bit.ly/32KF7zS)

Question #:8

Which of the following code blocks performs a join in which the small DataFrame transactionsDf is sent to all executors where it is joined with DataFrame itemsDf on columns storeId and itemId,

respectively?

A. itemsDf.join(transactionsDf, itemsDf.itemId == transactionsDf.storeId, "right_outer")

B. itemsDf.join(transactionsDf, itemsDf.itemId == transactionsDf.storeId, "broadcast")

C. itemsDf.merge(transactionsDf, "itemsDf.itemId == transactionsDf.storeId", "broadcast")

D. itemsDf.join(broadcast(transactionsDf), itemsDf.itemId == transactionsDf.storeId)

E. itemsDf.join(transactionsDf, broadcast(itemsDf.itemId == transactionsDf.storeId))

Answer: D

Explanation

Explanation

The issue with all answers that have "broadcast" as very last argument is that "broadcast" is not a valid join type. While the entry with "right_outer" is a valid statement, it is not a broadcast join. The

item where broadcast() is wrapped around the equality condition is not valid code in Spark. broadcast() needs to be wrapped around the name of the small DataFrame that should be broadcast.

More info: Learning Spark, 2nd Edition, Chapter 7

Static notebook | Dynamic notebook: See test 1, QUESTION NO: 34 (Databricks import instructions)

tion and explanation?

Question #:9

Which of the following code blocks applies the boolean-returning Python function evaluateTestSuccess to column storeId of DataFrame transactionsDf as a user-defined function?

A. 1.from pyspark.sql import types as T

2.evaluateTestSuccessUDF = udf(evaluateTestSuccess, T.BooleanType())

3.transactionsDf.withColumn("result", evaluateTestSuccessUDF(col("storeId")))

B. 1.evaluateTestSuccessUDF = udf(evaluateTestSuccess)

2.transactionsDf.withColumn("result", evaluateTestSuccessUDF(storeId))

C. 1.from pyspark.sql import types as T

2.evaluateTestSuccessUDF = udf(evaluateTestSuccess, T.IntegerType())

3.transactionsDf.withColumn("result", evaluateTestSuccess(col("storeId")))

D. 1.evaluateTestSuccessUDF = udf(evaluateTestSuccess)

2.transactionsDf.withColumn("result", evaluateTestSuccessUDF(col("storeId")))

E. 1.from pyspark.sql import types as T

2.evaluateTestSuccessUDF = udf(evaluateTestSuccess, T.BooleanType())

3.transactionsDf.withColumn("result", evaluateTestSuccess(col("storeId")))

Answer: A

Explanation

Explanation

Recognizing that the UDF specification requires a return type (unless it is a string, which is the default) is important for solving this question. In addition, you should make sure that the generated

UDF (evaluateTestSuccessUDF) and not the Python function (evaluateTestSuccess) is applied to column storeId.

More info: pyspark.sql.functions.udf — PySpark 3.1.2 documentation

Static notebook | Dynamic notebook: See test 2, QUESTION NO: 34 (Databricks import instructions)

Question #:10

Which of the following statements about Spark's execution hierarchy is correct?

A. In Spark's execution hierarchy, a job may reach over multiple stage boundaries.

B. In Spark's execution hierarchy, manifests are one layer above jobs.

C. In Spark's execution hierarchy, a stage comprises multiple jobs.

D. In Spark's execution hierarchy, executors are the smallest unit.

E. In Spark's execution hierarchy, tasks are one layer above slots.

Answer: A

Explanation

Explanation

In Spark's execution hierarchy, a job may reach over multiple stage boundaries.

Correct. A job is a sequence of stages, and thus may reach over multiple stage boundaries.

In Spark's execution hierarchy, tasks are one layer above slots.

Incorrect. Slots are not a part of the execution hierarchy. Tasks are the lowest layer.

In Spark's execution hierarchy, a stage comprises multiple jobs.

No. It is the other way around – a job consists of one or multiple stages.

In Spark's execution hierarchy, executors are the smallest unit.

False. Executors are not a part of the execution hierarchy. Tasks are the smallest unit!

In Spark's execution hierarchy, manifests are one layer above jobs.

Wrong. Manifests are not a part of the Spark ecosystem.

About dumpscafe.com

dumpscafe.com was founded in 2007. We provide latest & high quality IT / Business Certification Training Exam Questions, Study Guides, Practice Tests.

We help you pass any IT / Business Certification Exams with 100% Pass Guaranteed or Full Refund. Especially Cisco, CompTIA, Citrix, EMC, HP, Oracle, VMware, Juniper, Check Point, LPI, Nortel, EXIN and so on.

View list of all certification exams: All vendors

We prepare state-of-the art practice tests for certification exams. You can reach us at any of the email addresses listed below.

- Sales: sales@dumpscafe.com
- Feedback: feedback@dumpscafe.com
- Support: support@dumpscafe.com

Any problems about IT certification or our products, You can write us back and we will get back to you within 24 hours.