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 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.

Correct Answer: A

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.

QUESTION 2

In which order should the code blocks shown below be run in order to create a table of all values in column attributes next to the respective values in column supplier in DataFrame itemsDf?

1.

```
itemsDf.createOrReplaceView("itemsDf")
```

2.

```
spark.sql("FROM itemsDf SELECT \\'supplier\\', explode(\\'Attributes\\')")
```

3.

```
spark.sql("FROM itemsDf SELECT supplier, explode(attributes)")
```

4.

```
itemsDf.createOrReplaceTempView("itemsDf")
```

A. 4, 3

B. 1, 3

C. 2

D. 4, 2

E. 1, 2

Correct Answer: A

Static notebook | Dynamic notebook: See test 1, 56 (Databricks import instructions)

QUESTION 3

Which of the following code blocks returns all unique values across all values in columns value and productId in DataFrame transactionsDf in a one-column DataFrame?

A. tranactionsDf.select(\\'value\\').join(transactionsDf.select(\\'productId\\'), col(\\'value\\')==col(\\'productId\\'), \\'outer\\')

B. transactionsDf.select(col(\\'value\\'), col(\\'productId\\')).agg({\\'*\\': \\'count\\'})

C. transactionsDf.select(\\'value\\', \\'productId\\').distinct()

D. transactionsDf.select(\\'value\\').union(transactionsDf.select(\\'productId\\')).distinct()

E. transactionsDf.agg({\\'value\\': \\'collect_set\\', \\'productId\\': \\'collect_set\\'})

Correct Answer: D

QUESTION 4

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

QUESTION 5

Which of the following statements about Spark\\'s DataFrames is incorrect?

A. Spark\\'s DataFrames are immutable.

B. Spark\\'s DataFrames are equal to Python\\'s DataFrames.

C. Data in DataFrames is organized into named columns.

D. RDDs are at the core of DataFrames.

E. The data in DataFrames may be split into multiple chunks.

Correct Answer: B

Spark\\'s DataFrames are equal to Python\\'s or R\\'s DataFrames. No, they are not equal. They are only similar. A major difference between Spark and Python is that Spark\\'s DataFrames are distributed, whereby Python\\'s are not.

QUESTION 6

Which of the following code blocks concatenates rows of DataFrames transactionsDf and transactionsNewDf, omitting any duplicates?

A. transactionsDf.concat(transactionsNewDf).unique()

B. transactionsDf.union(transactionsNewDf).distinct()

C. spark.union(transactionsDf, transactionsNewDf).distinct()

D. transactionsDf.join(transactionsNewDf, how="union").distinct()

E. transactionsDf.union(transactionsNewDf).unique()

Correct Answer: B

DataFrame.unique() and DataFrame.concat() do not exist and union() is not a method of the SparkSession. In addition, there is no union option for the join method in the DataFrame.join() statement.

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

Static notebook | Dynamic notebook: See test 2, 43 (Databricks import instructions)

QUESTION 7

The code block displayed below contains multiple errors. The code block should remove column transactionDate from DataFrame transactionsDf and add a column transactionTimestamp in which

dates that are expressed as strings in column transactionDate of DataFrame transactionsDf are converted into unix timestamps. Find the errors.

Sample of DataFrame transactionsDf:

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

2.|transactionId|predError|value|storeId|productId| f| transactionDate|
3.+----------+---------+-----+-------+---------+----+----------+

4.| 1| 3| 4| 25| 1|null|2020-04-26 15:35|

5.| 2| 6| 7| 2| 2|null|2020-04-13 22:01|

6.| 3| 3| null| 25| 3|null|2020-04-02 10:53|

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

Code block:

1.transactionsDf = transactionsDf.drop("transactionDate")

2.transactionsDf["transactionTimestamp"] = unix_timestamp("transactionDate", "yyyy-MM- dd")

A. Column transactionDate should be dropped after transactionTimestamp has been written. The string indicating the date format should be adjusted. The withColumn operator should be used instead of the existing column assignment. Operator to_unixtime() should be used instead of unix_timestamp().

B. Column transactionDate should be dropped after transactionTimestamp has been written. The withColumn operator should be used instead of the existing column assignment. Column transactionDate should be wrapped in a col() operator.

C. Column transactionDate should be wrapped in a col() operator.

D. The string indicating the date format should be adjusted. The withColumnReplaced operator should be used instead of the drop and assign pattern in the code block to replace column transactionDate with the new column transactionTimestamp.

E. Column transactionDate should be dropped after transactionTimestamp has been written. The string indicating the date format should be adjusted. The withColumn operator should be used instead of the existing column assignment.

Correct Answer: E

This requires a lot of thinking to get right. For solving it, you may take advantage of the digital notepad that is provided to you during the test. You have probably seen that the code block includes multiple errors. In the test, you are usually confronted with a code block that only contains a single error. However, since you are practicing here, this challenging multi-error will make it easier for you to deal with single-error questions in the real exam. You can clearly see that column transactionDate should be dropped only after transactionTimestamp has been written. This is because to generate column transactionTimestamp, Spark needs to read the values from column transactionDate. Values in column transactionDate in the original transactionsDf DataFrame look like 2020- 04-26 15:35. So, to convert those correctly, you would have to pass yyyy-MM-dd HH:mm. In other words: The string indicating the date format should be adjusted. While you might be tempted to change unix_timestamp() to to_unixtime() (in line with the from_unixtime() operator), this function does not exist in Spark. unix_timestamp() is the correct operator to use here. Also, there is no DataFrame.withColumnReplaced() operator. A similar operator that exists is DataFrame.withColumnRenamed(). Whether you use col() or not is irrelevant with unix_timestamp() - the command is fine with both. Finally, you cannot assign a column like transactionsDf["columnName"] = ... in Spark. This is Pandas syntax (Pandas is a popular Python package for data analysis), but it is not supported in Spark. So, you need to use Spark\\'s DataFrame.withColumn() syntax instead. More info: pyspark.sql.functions.unix_timestamp -- PySpark 3.1.2 documentation Static notebook | Dynamic notebook: See test 3, 28 (Databricks import instructions)

QUESTION 8

The code block shown below should return only the average prediction error (column predError) of a random subset, without replacement, of approximately 15% of rows in DataFrame transactionsDf. Choose the answer that correctly fills the blanks in the code block to accomplish this.

transactionsDf.__1__(__2__, __3__).__4__(avg(\\'predError\\'))

A. 1. sample

2.

True

3.

0.15

4.

filter

B. 1. sample

2.

False

3.

0.15

4.

select

C. 1. sample

2.

0.85

3.

False

4.

select

D. 1. fraction

2.

0.15

3.

True

4.

where

E. 1. fraction

2.

False

3.

0.85

4.

select

Correct Answer: B

Correct code block: transactionsDf.sample(withReplacement=False, fraction=0.15).select(avg(\\'predError\\')) You should remember that getting a random subset of rows means sampling. This, in turn should point you to the DataFrame.sample() method. Once you know this, you can look up the correct order of arguments in the documentation (link below). Lastly, you have to decide whether to use filter, where or select. where is just an alias for filter(). filter() is not the correct method to use here, since it would only allow you to filter rows based on some condition. However, the

asks to return only the average prediction error. You can control the columns that a query returns with the select() method ?so this is the correct method to use here. More info: pyspark.sql.DataFrame.sample -- PySpark 3.1.2 documentation Static notebook | Dynamic notebook: See test 2, 28 (Databricks import instructions)

QUESTION 9

Which of the following is a problem with using accumulators?

A. Only unnamed accumulators can be inspected in the Spark UI.

B. Only numeric values can be used in accumulators.

C. Accumulator values can only be read by the driver, but not by executors.

D. Accumulators do not obey lazy evaluation.

E. Accumulators are difficult to use for debugging because they will only be updated once, independent if a task has to be re-run due to hardware failure.

Correct Answer: C

QUESTION 10

Which of the following describes characteristics of the Dataset API?

A. The Dataset API does not support unstructured data.

B. In Python, the Dataset API mainly resembles Pandas\\' DataFrame API.

C. In Python, the Dataset API\\'s schema is constructed via type hints.

D. The Dataset API is available in Scala, but it is not available in Python.

E. The Dataset API does not provide compile-time type safety.

Correct Answer: D

QUESTION 11

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)),

D. StructType([T.StructField("season", T.CharType()), T.StructField("season",

E. DoubleType())]))

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

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

Correct Answer: B

QUESTION 12

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 13

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 14

The code block displayed below contains an error. The code block should merge the rows of DataFrames transactionsDfMonday and transactionsDfTuesday into a new DataFrame, matching column names and inserting null values where column names do not appear in both DataFrames. Find the error.

Sample of DataFrame transactionsDfMonday:

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

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

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

4.| 5| null| null| null| 2|null|

5.| 6| 3| 2| 25| 2|null|

6.+----------+---------+-----+-------+---------+----+

Sample of DataFrame transactionsDfTuesday:

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

2.|storeId|transactionId|productId|value|

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

4.| 25| 1| 1| 4|

5.| 2| 2| 2| 7|

6.| 3| 4| 2| null|

7.| null| 5| 2| null|

8.+-------+-------------+---------+-----+

Code block:

sc.union([transactionsDfMonday, transactionsDfTuesday])

A. The DataFrames\\' RDDs need to be passed into the sc.union method instead of the DataFrame variable names.

B. Instead of union, the concat method should be used, making sure to not use its default arguments.

C. Instead of the Spark context, transactionDfMonday should be called with the join method instead of the union method, making sure to use its default arguments.

D. Instead of the Spark context, transactionDfMonday should be called with the union method.

E. Instead of the Spark context, transactionDfMonday should be called with the unionByName method instead of the union method, making sure to not use its default arguments.

Correct Answer: E

Correct code block:

transactionsDfMonday.unionByName(transactionsDfTuesday, True) Output of correct code block:

+-------------+---------+-----+-------+---------+----+ |transactionId|predError|value|storeId|productId| f|

+-------------+---------+-----+-------+---------+----+ | 5| null| null| null| 2|null|

| 6| 3| 2| 25| 2|null|

| 1| null| 4| 25| 1|null|

| 2| null| 7| 2| 2|null|

| 4| null| null| 3| 2|null|

| 5| null| null| null| 2|null|

+-------------+---------+-----+-------+---------+----+ For solving this question, you should be aware of the

difference between the DataFrame.union() and DataFrame.unionByName() methods. The first one

matches columns independent of their

names, just by their order. The second one matches columns by their name (which is asked for in the

QUESTION 15

Which of the following code blocks returns a DataFrame that has all columns of DataFrame transactionsDf and an additional column predErrorSquared which is the squared value of column predError in DataFrame transactionsDf?

A. transactionsDf.withColumn("predError", pow(col("predErrorSquared"), 2))

B. transactionsDf.withColumnRenamed("predErrorSquared", pow(predError, 2))

C. transactionsDf.withColumn("predErrorSquared", pow(col("predError"), lit(2)))

D. transactionsDf.withColumn("predErrorSquared", pow(predError, lit(2)))

E. transactionsDf.withColumn("predErrorSquared", "predError"**2)

Correct Answer: C