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.leads4pass.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 RDDs is incorrect?

A. An RDD consists of a single partition.

B. The high-level DataFrame API is built on top of the low-level RDD API.

C. RDDs are immutable.

D. RDD stands for Resilient Distributed Dataset.

E. RDDs are great for precisely instructing Spark on how to do a query.

Correct Answer: A

QUESTION 2

The code block shown below should add column transactionDateForm to DataFrame transactionsDf. The column should express the unix-format timestamps in column transactionDate as string type like Apr 26 (Sunday). Choose the answer that correctly fills the blanks in the code block to accomplish this.

transactionsDf.__1__(__2__, from_unixtime(__3__, __4__))

A. 1. withColumn

2.

"transactionDateForm"

3.

"MMM d (EEEE)"

4.

"transactionDate"

B. 1. select

2.

"transactionDate"

3.

"transactionDateForm"

4.

"MMM d (EEEE)"

C. 1. withColumn

2.

"transactionDateForm"

3.

"transactionDate"

4.

"MMM d (EEEE)"

D. 1. withColumn

2.

"transactionDateForm"

3.

"transactionDate"

4.

"MM d (EEE)"

E. 1. withColumnRenamed

2.

"transactionDate"

3.

"transactionDateForm"

4.

"MM d (EEE)"

Correct Answer: C

QUESTION 3

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 4

Which of the following code blocks returns a one-column DataFrame of all values in column supplier of DataFrame itemsDf that do not contain the letter X? In the DataFrame, every value should only be listed once.

Sample of DataFrame itemsDf:

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

2.|itemId| itemName| attributes| supplier|

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

4.| 1|Thick Coat for Wa...|[blue, winter, cozy]|Sports Company Inc.|

5.| 2|Elegant Outdoors ...|[red, summer, fre...| YetiX|

6.| 3| Outdoors Backpack|[green, summer, t...|Sports Company Inc.|

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

A. itemsDf.filter(col(supplier).not_contains(\\'X\\')).select(supplier).distinct()

B. itemsDf.select(~col(\\'supplier\\').contains(\\'X\\')).distinct()

C. itemsDf.filter(not(col(\\'supplier\\').contains(\\'X\\'))).select(\\'supplier\\').unique()

D. itemsDf.filter(~col(\\'supplier\\').contains(\\'X\\')).select(\\'supplier\\').distinct()

E. itemsDf.filter(!col(\\'supplier\\').contains(\\'X\\')).select(col(\\'supplier\\')).unique()

Correct Answer: D

Output of correct code block:

+--------------------+

| supplier|

+-------------------+

|Sports Company Inc.|

+-------------------+

Key to managing this is understand which operator to use to do the opposite of an operation ?the ~ (not)

operator. In addition, you should know that there is no unique() method.

Static notebook | Dynamic notebook: See test 1, 55 (Databricks import instructions)

QUESTION 5

Which of the following is the idea behind dynamic partition pruning in Spark?

A. Dynamic partition pruning is intended to skip over the data you do not need in the results of a query.

B. Dynamic partition pruning concatenates columns of similar data types to optimize join performance.

C. Dynamic partition pruning performs wide transformations on disk instead of in memory.

D. Dynamic partition pruning reoptimizes physical plans based on data types and broadcast variables.

E. Dynamic partition pruning reoptimizes query plans based on runtime statistics collected during query execution.

Correct Answer: A

QUESTION 6

Which of the following code blocks returns a copy of DataFrame transactionsDf that only includes columns

transactionId, storeId, productId and f?

Sample of 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.+-------------+---------+-----+-------+---------+----+

A. transactionsDf.drop(col("value"), col("predError"))

B. transactionsDf.drop("predError", "value")

C. transactionsDf.drop(value, predError)

D. transactionsDf.drop(["predError", "value"])

E. transactionsDf.drop([col("predError"), col("value")])

Correct Answer: B

QUESTION 7

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 8

Which of the following describes the conversion of a computational query into an execution plan in Spark?

A. Spark uses the catalog to resolve the optimized logical plan.

B. The catalog assigns specific resources to the optimized memory plan.

C. The executed physical plan depends on a cost optimization from a previous stage.

D. Depending on whether DataFrame API or SQL API are used, the physical plan may differ.

E. The catalog assigns specific resources to the physical plan.

Correct Answer: C

The executed physical plan depends on a cost optimization from a previous stage. Correct! Spark considers multiple

physical plans on which it performs a cost analysis and selects the final physical plan in accordance with the lowest-cost outcome of that analysis. That final physical plan is then executed by Spark. Spark uses the catalog to resolve the optimized logical plan. No. Spark uses the catalog to resolve the unresolved logical plan, but not the optimized logical plan. Once the unresolved logical plan is resolved, it is then optimized using the Catalyst Optimizer. The optimized logical plan is the input for physical planning. The catalog assigns specific resources to the physical plan. No. The catalog stores metadata, such as a list of names of columns, data types, functions, and databases. Spark consults the catalog for resolving the references in a logical plan at the beginning of the conversion of the query into an execution plan. The result is then an optimized logical plan. Depending on whether DataFrame API or SQL API are used, the physical plan may differ. Wrong ?the physical plan is independent of which API was used. And this is one of the great strengths of Spark! The catalog assigns specific resources to the optimized memory plan. There is no specific "memory plan" on the journey of a Spark computation. More info: Spark\\'s Logical and Physical plans ... When, Why, How and Beyond. | by Laurent Leturgez | datalex | Medium

---

QUESTION 9

Which of the following describes characteristics of the Spark driver?

A. The Spark driver requests the transformation of operations into DAG computations from the worker nodes.

B. If set in the Spark configuration, Spark scales the Spark driver horizontally to improve parallel processing performance.

C. The Spark driver processes partitions in an optimized, distributed fashion.

D. In a non-interactive Spark application, the Spark driver automatically creates the SparkSession object.

E. The Spark driver\\'s responsibility includes scheduling queries for execution on worker nodes.

Correct Answer: D

The Spark driver requests the transformation of operations into DAG computations from the worker nodes.

No, the Spark driver transforms operations into DAG computations itself. If set in the Spark configuration,

Spark scales the Spark driver horizontally to improve parallel processing performance.

No. There is always a single driver per application, but one or more executors. The Spark driver processes

partitions in an optimized, distributed fashion.

No, this is what executors do.

In a non-interactive Spark application, the Spark driver automatically creates the SparkSession object.

Wrong. In a non-interactive Spark application, you need to create the SparkSession object. In an

interactive Spark shell, the Spark driver instantiates the object for you.

QUESTION 10

Which of the following statements about executors is correct?

A. Executors are launched by the driver.

B. Executors stop upon application completion by default.

C. Each node hosts a single executor.

D. Executors store data in memory only.

E. An executor can serve multiple applications.

Correct Answer: B

QUESTION 11

Which of the following code blocks returns a copy of DataFrame itemsDf where the column supplier has been renamed to manufacturer?

A. itemsDf.withColumn(["supplier", "manufacturer"])

B. itemsDf.withColumn("supplier").alias("manufacturer")

C. itemsDf.withColumnRenamed("supplier", "manufacturer")

D. itemsDf.withColumnRenamed(col("manufacturer"), col("supplier"))

E. itemsDf.withColumnsRenamed("supplier", "manufacturer")

Correct Answer: C

itemsDf.withColumnRenamed("supplier", "manufacturer") Correct! This uses the relatively trivial

DataFrame method withColumnRenamed for renaming column supplier to column manufacturer.

Note that the asks for "a copy of DataFrame itemsDf". This may be confusing if you are not familiar with

Spark yet. RDDs (Resilient Distributed Datasets) are the foundation of

Spark DataFrames and are immutable. As such, DataFrames are immutable, too. Any command that

changes anything in the DataFrame therefore necessarily returns a copy, or a new version, of it

that has the changes applied.

itemsDf.withColumnsRenamed("supplier", "manufacturer") Incorrect. Spark\\'s DataFrame API does not

have a withColumnsRenamed() method. itemsDf.withColumnRenamed(col("manufacturer"), col

("supplier")) No. Watch out ?although the col() method works for many methods of the DataFrame API,

withColumnRenamed is not one of them. As outlined in the documentation linked below,

withColumnRenamed expects strings.

itemsDf.withColumn(["supplier", "manufacturer"])

Wrong. While DataFrame.withColumn() exists in Spark, it has a different purpose than renaming columns.

withColumn is typically used to add columns to DataFrames, taking the name of the new

column as a first, and a Column as a second argument. Learn more via the documentation that is linked

below.

itemsDf.withColumn("supplier").alias("manufacturer") No. While DataFrame.withColumn() exists, it

requires 2 arguments. Furthermore, the alias() method on DataFrames would not help the cause of

renaming a column much.

DataFrame.alias() can be

useful in addressing the input of join statements. However, this is far outside of the scope of this question.

If you are curious nevertheless, check out the link below. More info:

pyspark.sql.DataFrame.withColumnRenamed -- PySpark 3.1.1 documentation,

pyspark.sql.DataFrame.withColumn -- PySpark 3.1.1 documentation, and pyspark.sql.DataFrame.alias --

PySpark 3.1.2 documentation (https://bit.ly/3aSB5tm , https://bit.ly/2Tv4rbE , https://bit.ly/2RbhBd2)

Static notebook | Dynamic notebook: See test 1, 31 (Databricks import instructions) (https://flrs.github.io/

spark_practice_tests_code/#1/31.html , https://bit.ly/sparkpracticeexams_import_instructions)

QUESTION 12

Which of the following is one of the big performance advantages that Spark has over Hadoop?

A. Spark achieves great performance by storing data in the DAG format, whereas Hadoop can only use parquet files.

B. Spark achieves higher resiliency for queries since, different from Hadoop, it can be deployed on Kubernetes.

C. Spark achieves great performance by storing data and performing computation in memory, whereas large jobs in Hadoop require a large amount of relatively slow disk I/O operations.

D. Spark achieves great performance by storing data in the HDFS format, whereas Hadoop can only use parquet files.

E. Spark achieves performance gains for developers by extending Hadoop\\'s DataFrames with a user-friendly API.

Correct Answer: C

QUESTION 13

Which of the following code blocks returns approximately 1000 rows, some of them potentially being duplicates, from the 2000-row DataFrame transactionsDf that only has unique rows?

A. transactionsDf.sample(True, 0.5)

B. transactionsDf.take(1000).distinct()

C. transactionsDf.sample(False, 0.5)

D. transactionsDf.take(1000)

E. transactionsDf.sample(True, 0.5, force=True)

Correct Answer: A

To solve this question, you need to know that DataFrame.sample() is not guaranteed to return the exact fraction of the number of rows specified as an argument. Furthermore, since duplicates may be returned, you should understand that the operator\\'s withReplacement argument should be set to True. A force= argument for the operator does not exist. While the take argument returns an exact number of rows, it will just take the first specified number of rows (1000 in this question) from the DataFrame. Since the DataFrame does not include duplicate rows, there is no potential of any of those returned rows being duplicates when using take(), so the correct answer cannot involve take().

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

Static notebook | Dynamic notebook: See test 2, 41 (Databricks import instructions)

QUESTION 14

Which of the following code blocks writes DataFrame itemsDf to disk at storage location filePath, making sure to substitute any existing data at that location?

A. itemsDf.write.mode("overwrite").parquet(filePath)

B. itemsDf.write.option("parquet").mode("overwrite").path(filePath)

C. itemsDf.write(filePath, mode="overwrite")

D. itemsDf.write.mode("overwrite").path(filePath)

E. itemsDf.write().parquet(filePath, mode="overwrite")

Correct Answer: A

QUESTION 15

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.