Up-to-date Practice Test with Latest Questions and Answers covering latest syllabus and topics of the exam. Makes you ready to face actual exam.

DCAD Practice Questions
DCAD Practice Test
DCAD Practice Exam
DCAD Exam Questions
DCAD Study Guide

killexams.com

Databricks

DCAD

Databricks Certified Associate Developer for Apache Spark 3.0

ORDER FULL VERSION

https://killexams.com/pass4sure/exam-detail/DCAD

Question: 386

Which of the following code blocks removes all rows in the 6-column DataFrame transactionsDf that have missing data in at least 3 columns?
A. transactionsDf.dropna("any")
B. transactionsDf.dropna(thresh=4)
C. transactionsDf.drop.na("",2)
D. transactionsDf.dropna(thresh=2)
E. transactionsDf.dropna("",4)

Answer: B

Explanation:

transactionsDf.dropna(thresh=4)

Correct. Note that by only working with the thresh keyword argument, the first how keyword argument is ignored. Also, figuring out which value to set for thresh can be difficult, especially when

under pressure in the exam. Here, I recommend you use the notes to create a "simulation" of what different values for thresh would do to a DataFrame. Here is an explanatory image why thresh=4 is

the correct answer to the question:

transactionsDf.dropna(thresh=2)

Almost right. See the comment about thresh for the correct answer above. transactionsDf.dropna("any")

No, this would remove all rows that have at least one missing value.

transactionsDf.drop.na("",2)

No, drop.na is not a proper DataFrame method.

transactionsDf.dropna("",4)

No, this does not work and will throw an error in Spark because Spark cannot understand the first argument.

More info: pyspark.sql.DataFrame.dropna — PySpark 3.1.1 documentation (https://bit.ly/2QZpiCp)

Static notebook | Dynamic notebook: See test 1,

Question: 387

"left_semi"

Answer: C

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: 388

Which of the elements that are labeled with a circle and a number contain an error or are misrepresented?
A. 1, 10
B. 1, 8
C. 10
D. 7, 9, 10
E. 1, 4, 6, 9

Answer: B

Explanation:

1: Correct C This should just read "API" or "DataFrame API". The DataFrame is not part of the SQL API. To make a DataFrame accessible via SQL, you first need to create a DataFrame view. That view can then be accessed via SQL.

4: Although "K_38_INU" looks odd, it is a completely valid name for a DataFrame column.

6: No, StringType is a correct type.

7: Although a StringType may not be the most efficient way to store a phone number, there is nothing fundamentally wrong with using this type here.

8: Correct C TreeType is not a type that Spark supports.

9: No, Spark DataFrames support ArrayType variables. In this case, the variable would represent a sequence of elements with type LongType, which is also a valid type for Spark DataFrames.

10: There is nothing wrong with this row.

More info: Data Types – Spark 3.1.1 Documentation (https://bit.ly/3aAPKJT)

Question: 389

Which of the following code blocks stores DataFrame itemsDf in executor memory and, if insufficient memory is available, serializes it and saves it to disk?
A. itemsDf.persist(StorageLevel.MEMORY_ONLY)
B. itemsDf.cache(StorageLevel.MEMORY_AND_DISK)
C. itemsDf.store()
D. itemsDf.cache()
E. itemsDf.write.option('destination', 'memory').save()

Answer: D

Explanation:

The key to solving this QUESTION NO: is knowing (or reading in the documentation) that, by default, cache() stores values to memory and writes any partitions for which there is insufficient memory

to disk. persist() can achieve the exact same behavior, however not with the StorageLevel.MEMORY_ONLY option listed here. It is also worth noting that cache() does not have any arguments.

If you have troubles finding the storage level information in the documentation, please also see this student Q&A thread that sheds some light here.

Static notebook | Dynamic notebook: See test 2,

Question: 390

Which of the following code blocks can be used to save DataFrame transactionsDf to memory only, recalculating partitions that do not fit in memory when they are needed?
A. from pyspark import StorageLevel transactionsDf.cache(StorageLevel.MEMORY_ONLY)
B. transactionsDf.cache()
C. transactionsDf.storage_level('MEMORY_ONLY')
D. transactionsDf.persist()
E. transactionsDf.clear_persist()
F. from pyspark import StorageLevel transactionsDf.persist(StorageLevel.MEMORY_ONLY)

Answer: F

Explanation:

from pyspark import StorageLevel transactionsDf.persist(StorageLevel.MEMORY_ONLY) Correct. Note that the storage level MEMORY_ONLY means that all partitions that do not fit into memory will be recomputed when they are needed. transactionsDf.cache()

This is wrong because the default storage level of DataFrame.cache() is

MEMORY_AND_DISK, meaning that partitions that do not fit into memory are stored on disk.

transactionsDf.persist()

This is wrong because the default storage level of DataFrame.persist() is

MEMORY_AND_DISK.

transactionsDf.clear_persist()

Incorrect, since clear_persist() is not a method of DataFrame.

transactionsDf.storage_level('MEMORY_ONLY')

Wrong. storage_level is not a method of DataFrame.

More info: RDD Programming Guide – Spark 3.0.0 Documentation, pyspark.sql.DataFrame.persist — PySpark 3.0.0 documentation (https://bit.ly/3sxHLVC , https://bit.ly/3j2N6B9)

Question: 391

"left_semi"

Answer: C

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: 392

Which of the following describes tasks?
A. A task is a command sent from the driver to the executors in response to a transformation.
B. Tasks transform jobs into DAGs.
C. A task is a collection of slots.
D. A task is a collection of rows.
E. Tasks get assigned to the executors by the driver.

Answer: E

Explanation:

Tasks get assigned to the executors by the driver.

Correct! Or, in other words: Executors take the tasks that they were assigned to by the driver, run them over partitions, and report the their outcomes back to the driver. Tasks transform jobs into DAGs.

No, this statement disrespects the order of elements in the Spark hierarchy. The Spark driver transforms jobs into DAGs. Each job consists of one or more stages. Each stage contains one or more

tasks.

A task is a collection of rows.

Wrong. A partition is a collection of rows. Tasks have little to do with a collection of rows. If anything, a task processes a specific partition.

A task is a command sent from the driver to the executors in response to a transformation. Incorrect. The Spark driver does not send anything to the executors in response to a transformation, since transformations are evaluated lazily. So, the Spark driver would send tasks to executors

only in response to actions.

A task is a collection of slots.

No. Executors have one or more slots to process tasks and each slot can be assigned a task.

Question: 393

Which of the following code blocks reads in parquet file /FileStore/imports.parquet as a

DataFrame?
A. spark.mode("parquet").read("/FileStore/imports.parquet")
B. spark.read.path("/FileStore/imports.parquet", source="parquet")
C. spark.read().parquet("/FileStore/imports.parquet")
D. spark.read.parquet("/FileStore/imports.parquet")

E. spark.read().format(‘parquet’).open("/FileStore/imports.parquet")

Answer: D

Explanation:

Static notebook | Dynamic notebook: See test 1,

Question: 394

Which of the elements that are labeled with a circle and a number contain an error or are misrepresented?
A. 1, 10
B. 1, 8
C. 10
D. 7, 9, 10
E. 1, 4, 6, 9

Answer: B

Explanation:

1: Correct C This should just read "API" or "DataFrame API". The DataFrame is not part of the SQL API. To make a DataFrame accessible via SQL, you first need to create a DataFrame view. That view can then be accessed via SQL.

4: Although "K_38_INU" looks odd, it is a completely valid name for a DataFrame column.

6: No, StringType is a correct type.

7: Although a StringType may not be the most efficient way to store a phone number, there is nothing fundamentally wrong with using this type here.

8: Correct C TreeType is not a type that Spark supports.

9: No, Spark DataFrames support ArrayType variables. In this case, the variable would represent a sequence of elements with type LongType, which is also a valid type for Spark DataFrames.

10: There is nothing wrong with this row.

More info: Data Types – Spark 3.1.1 Documentation (https://bit.ly/3aAPKJT)

Question: 395

"left_semi"

Answer: C

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: 396

Which of the elements that are labeled with a circle and a number contain an error or are misrepresented?
A. 1, 10
B. 1, 8
C. 10
D. 7, 9, 10
E. 1, 4, 6, 9

Answer: B

Explanation:

1: Correct C This should just read "API" or "DataFrame API". The DataFrame is not part of the SQL API. To make a DataFrame accessible via SQL, you first need to create a DataFrame view. That view can then be accessed via SQL.

4: Although "K_38_INU" looks odd, it is a completely valid name for a DataFrame column.

6: No, StringType is a correct type.

7: Although a StringType may not be the most efficient way to store a phone number, there is nothing fundamentally wrong with using this type here.

8: Correct C TreeType is not a type that Spark supports.

9: No, Spark DataFrames support ArrayType variables. In this case, the variable would represent a sequence of elements with type LongType, which is also a valid type for Spark DataFrames.

10: There is nothing wrong with this row.

More info: Data Types – Spark 3.1.1 Documentation (https://bit.ly/3aAPKJT)

KILLEXAMS.COM

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including Exam Questions, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.

Practice Exam Questions Based on Current Exam Objectives

Killexams.com provides practice exam questions aligned with the latest official exam objectives and latest syllabus. Our content is reviewed and updated regularly to reflect recent changes announced by certification vendors. By studying these practice questions, candidates will cover the structure, difficulty level, and topics of the actual exam, helping them prepare more effectively and efficiently.

Comprehensive Practice Exams (PDF Format)

Killexams.com offers multiple-choice questions (MCQs) in easy-to-read PDF format, covering all major domains of the exam. Each PDF contains a structured collection of practice questions and verified answers designed to support focused study. These MCQs help candidates reinforce key concepts, identify knowledge gaps, and improve exam readiness through consistent practice.

Realistic Practice Tests (Online Test Engine & Desktop Test Engine)

To support hands-on preparation, Killexams.com provides practice tests through both an Online Test Engine and a Desktop Test Engine. These tools are designed to simulate a real exam environment, allowing candidates to practice under exam-like conditions, with latest syllbus and topics of the exam. Performance tracking, test history, and result analysis help users evaluate their progress and focus on areas that need improvement.

Risk-Free Purchase Policy

Killexams.com follows a transparent and customer-friendly purchase policy. If users are not satisfied with the study materials, they may request assistance or a refund in accordance with our published terms and conditions. This policy reflects our commitment to customer satisfaction, fairness, and confidence in our preparation resources.

Regularly Updated Content

Our practice question bank is reviewed and updated on an ongoing basis to stay aligned with the latest exam outlines and vendor updates. This ensures candidates are studying up-to-date, relevant material, and preparing with content that reflects current exam expectations, helping them stay confident and well-prepared.