transf	ormations like <b>join()</b> ?		
0	spark.shuffle.file.buffer		
0	spark.shuffle.io.maxRetries		
×	spark.sql.shuffle.partitions		
0	spark.default.parallelism		
0	spark.shuffle.partitions		
The code block shown below contains an error. The code block is intended to write DataFrame storesDF to file path filePath as parquet and partition by values in column division. Identify the error.  Code block:  storesDF.write.repartition("division").parquet(filePath)			
0	There is no parquet() operation for DataFrameWriter — the save() operation should be used instead.		
0	The mode() operation must be called to specify that this write should not overwrite existing files.		
×	There is no repartition() operation for DataFrameWriter — the partitionBy() operation should be used instead.		
0	DataFrame.write is an operation — it should be followed by parentheses to return a DataFrameWriter.		
Which	n of the following operations can be used to sort the rows of a DataFrame?		
Ø	sort() and orderBy()		
0	orderby()		
0	sort() and orderby()		
0	sort()		
0	orderBy()		

Which of the following Spark config properties represents the number of partitions used in wide

Which of the following code blocks returns a DataFrame where rows in DataFrame storesDF containing missing values in every column have been dropped?

StoresDF.dropna()

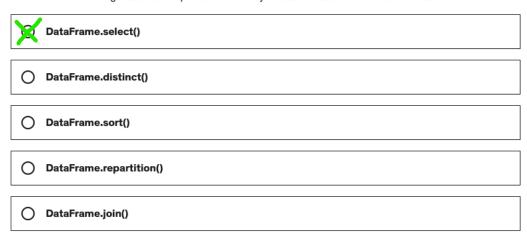
StoresDF.na.drop()

storesDF.na.drop("all")

storesDF.na.drop("all", subset = "sqft")

storesDF.nadrop("all")

Which of the following DataFrame operations is always classified as a narrow transformation?



Which of the following describes nodes in cluster-mode Spark?

- Nodes are the most granular level of execution in the Spark execution hierarchy.
- There are driver nodes and worker nodes, both of which can scale horizontally.
- Nodes are another term for executors, so they are processing engine instances for performing computations.
- There is only one node and it hosts both the driver and executors.

Worker nodes are machines that host the executors responsible for the execution of tasks.

Which of the following code blocks will **not always** return the exact number of distinct values in column division?

0	<pre>storesDF.select("division").dropDuplicates().count()</pre>
X	<pre>storesDF.agg(approx_count_distinct(col("division")).alias("divisionDistinct"))</pre>
0	<pre>storesDF.select("division").distinct().count()</pre>
0	<pre>storesDF.agg(countDistinct(col("division")).alias("divisionDistinct"))</pre>
0	<pre>storesDF.agg(approx_count_distinct(col("division"),0).alias("divisionDistinct"))</pre>
The co	de block shown below should return a new DataFrame that is the result of a cross join
betwe	en DataFrame <b>storesDF</b> and DataFrame <b>employeesDF</b> . Choose the response that
correc	tly fills in the numbered blanks within the code block to complete this task.
Code k	olock:
1_	2(3)
0	1. storesDF 2. join
	3. employeesDF, "cross"
0	1. storesDF 2. crossJoin 3. employeesDF, "storeId"
×	1. storesDF 2. crossJoin 3. employeesDF, "storeId"
0	1. storesDF 2. crossJoin 3. employeesDF
0	1. storesDF 2. join 3. employeesDF, "storeId", "cross"

Which of the following code blocks returns a new DataFrame from DataFrame **storesDF** where column **numberOfManagers** is the constant integer 1?

<pre>storesDF.withColumn("numberOfManagers", lit("1"))</pre>		
<pre>storesDF.withColumn("numberOfManagers", 1)</pre>		
<pre>storesDF.withColumn("numberOfManagers", IntegerType(1))</pre>		
storesDF.withColumn("numberOfManagers", lit(1))		
<pre>storesDF.withColumn("numberOfManagers", col(1))</pre>		
Which of the following code blocks returns a new DataFrame from DataFrame <b>storesDF</b> where column <b>numberOfManagers</b> is the constant integer 1?		
<pre>storesDF.withColumn("numberOfManagers", lit("1"))</pre>		
<pre>storesDF.withColumn("numberOfManagers", 1)</pre>		
·		
<pre>storesDF.withColumn("numberOfManagers", IntegerType(1))</pre>		
<pre>storesDF.withColumn("numberOfManagers", lit(1))</pre>		

storesDF.withColumn("numberOfManagers", col(1))