## **Topic 1 - Exam A**

Question #1*Topic 1*

Which of the following describes the Spark driver?

A. The Spark driver is responsible for performing all execution in all execution modes – it is the entire Spark application.

B. The Spare driver is fault tolerant – if it fails, it will recover the entire Spark application.

C. The Spark driver is the coarsest level of the Spark execution hierarchy – it is synonymous with the Spark application.

D. The Spark driver is the program space in which the Spark application’s main method runs coordinating the Spark entire application.

E. The Spark driver is horizontally scaled to increase overall processing throughput of a Spark application.

Question #2*Topic 1*

Which of the following describes the relationship between nodes and executors?

A. Executors and nodes are not related.

B. Anode is a processing engine running on an executor.

C. An executor is a processing engine running on a node.

D. There are always the same number of executors and nodes.

E. There are always more nodes than executors.

Question #3*Topic 1*

Which of the following will occur if there are more slots than there are tasks?

A. The Spark job will likely not run as efficiently as possible.

B. The Spark application will fail – there must be at least as many tasks as there are slots.

C. Some executors will shut down and allocate all slots on larger executors first.

D. More tasks will be automatically generated to ensure all slots are being used.

E. The Spark job will use just one single slot to perform all tasks.

Question #4*Topic 1*

Which of the following is the most granular level of the Spark execution hierarchy?

A. Task

B. Executor

C. Node

D. Job

E. Slot

Question #5*Topic 1*

Which of the following statements about Spark jobs is incorrect?

A. Jobs are broken down into stages.

B. There are multiple tasks within a single job when a DataFrame has more than one partition.

C. Jobs are collections of tasks that are divided up based on when an action is called.

D. There is no way to monitor the progress of a job.

E. Jobs are collections of tasks that are divided based on when language variables are defined.

Question #6*Topic 1*

Which of the following operations is most likely to result in a shuffle?

A. DataFrame.join()

B. DataFrame.filter()

C. DataFrame.union()

D. DataFrame.where()

E. DataFrame.drop()

Question #7*Topic 1*

The default value of spark.sql.shuffle.partitions is 200. Which of the following describes what that means?

A. By default, all DataFrames in Spark will be spit to perfectly fill the memory of 200 executors.

B. By default, new DataFrames created by Spark will be split to perfectly fill the memory of 200 executors.

C. By default, Spark will only read the first 200 partitions of DataFrames to improve speed.

D. By default, all DataFrames in Spark, including existing DataFrames, will be split into 200 unique segments for parallelization.

E. By default, DataFrames will be split into 200 unique partitions when data is being shuffled.

Question #8*Topic 1*

Which of the following is the most complete description of lazy evaluation?

A. None of these options describe lazy evaluation

B. A process is lazily evaluated if its execution does not start until it is put into action by some type of trigger

C. A process is lazily evaluated if its execution does not start until it is forced to display a result to the user

D. A process is lazily evaluated if its execution does not start until it reaches a specified date and time

E. A process is lazily evaluated if its execution does not start until it is finished compiling

Question #9*Topic 1*

Which of the following DataFrame operations is classified as an action?

A. DataFrame.drop()

B. DataFrame.coalesce()

C. DataFrame.take()

D. DataFrame.join()

E. DataFrame.filter()

Question #10*Topic 1*

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

A. DataFrame.filter()

B. DataFrame.join()

C. DataFrame.select()

D. DataFrame.drop()

E. DataFrame.union()

Question #11*Topic 1*

Which of the following describes the difference between cluster and client execution modes?

A. The cluster execution mode runs the driver on a worker node within a cluster, while the client execution mode runs the driver on the client machine (also known as a gateway machine or edge node).

B. The cluster execution mode is run on a local cluster, while the client execution mode is run in the cloud.

C. The cluster execution mode distributes executors across worker nodes in a cluster, while the client execution mode runs a Spark job entirely on one client machine.

D. The cluster execution mode runs the driver on the cluster machine (also known as a gateway machine or edge node), while the client execution mode runs the driver on a worker node within a cluster.

E. The cluster execution mode distributes executors across worker nodes in a cluster, while the client execution mode submits a Spark job from a remote machine to be run on a remote, unconfigurable cluster.

Question #12*Topic 1*

Which of the following statements about Spark’s stability is incorrect?

A. Spark is designed to support the loss of any set of worker nodes.

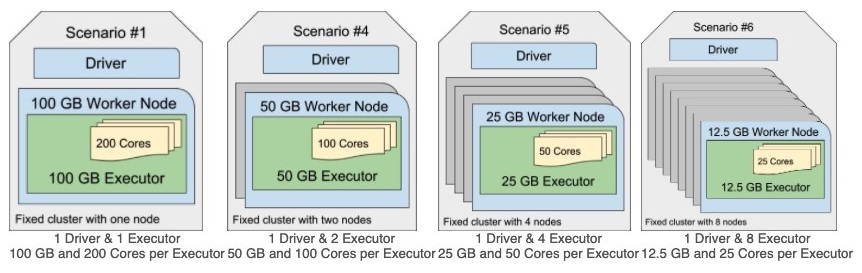
B. Spark will rerun any failed tasks due to failed worker nodes.

C. Spark will recompute data cached on failed worker nodes.

D. Spark will spill data to disk if it does not fit in memory.

E. Spark will reassign the driver to a worker node if the driver’s node fails.

Question #13*Topic 1*

Which of the following cluster configurations is most likely to experience an out-of-memory error in response to data skew in a single partition?  
  
Note: each configuration has roughly the same compute power using 100 GB of RAM and 200 cores.

A. Scenario #4

B. Scenario #5

C. Scenario #6

D. More information is needed to determine an answer.

E. Scenario #1

Question #14*Topic 1*

Of the following situations, in which will it be most advantageous to store DataFrame df at the MEMORY\_AND\_DISK storage level rather than the MEMORY\_ONLY storage level?

A. When all of the computed data in DataFrame df can fit into memory.

B. When the memory is full and it’s faster to recompute all the data in DataFrame df rather than read it from disk.

C. When it’s faster to recompute all the data in DataFrame df that cannot fit into memory based on its logical plan rather than read it from disk.

D. When it’s faster to read all the computed data in DataFrame df that cannot fit into memory from disk rather than recompute it based on its logical plan.

E. The storage level MENORY\_ONLY will always be more advantageous because it’s faster to read data from memory than it is to read data from disk.

Question #15*Topic 1*

A Spark application has a 128 GB DataFrame A and a 1 GB DataFrame B. If a broadcast join were to be performed on these two DataFrames, which of the following describes which DataFrame should be broadcasted and why?

A. Either DataFrame can be broadcasted. Their results will be identical in result and efficiency.

B. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself.

C. DataFrame A should be broadcasted because it is larger and will eliminate the need for the shuffling of DataFrame B.

D. DataFrame B should be broadcasted because it is smaller and will eliminate the need for the shuffling of DataFrame A.

E. DataFrame A should be broadcasted because it is smaller and will eliminate the need for the shuffling of itself.

Question #16*Topic 1*

Which of the following operations can be used to create a new DataFrame that has 12 partitions from an original DataFrame df that has 8 partitions?

A. df.repartition(12)

B. df.cache()

C. df.partitionBy(1.5)

D. df.coalesce(12)

E. df.partitionBy(12)

Question #17*Topic 1*

Which of the following object types cannot be contained within a column of a Spark DataFrame?

A. DataFrame

B. String

C. Array

D. null

E. Vector

Question #18*Topic 1*

Which of the following operations can be used to create a DataFrame with a subset of columns from DataFrame storesDF that are specified by name?

A. storesDF.subset()

B. storesDF.select()

C. storesDF.selectColumn()

D. storesDF.filter()

E. storesDF.drop()

Question #19*Topic 1*

The code block shown below contains an error. The code block is intended to return a DataFrame containing all columns from DataFrame storesDF except for column sqft and column customerSatisfaction. Identify the error.  
Code block:  
storesDF.drop(sqft, customerSatisfaction)

A. The drop() operation only works if one column name is called at a time – there should be two calls in succession like storesDF.drop("sqft").drop("customerSatisfaction").

B. The drop() operation only works if column names are wrapped inside the col() function like storesDF.drop(col(sqft), col(customerSatisfaction)).

C. There is no drop() operation for storesDF.

D. The sqft and customerSatisfaction column names should be quoted like "sqft" and "customerSatisfaction".

E. The sqft and customerSatisfaction column names should be subset from the DataFrame storesDF like storesDF."sqft" and storesDF."customerSatisfaction".

Question #20*Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000?

A. storesDF.filter("sqft" <= 25000)

B. storesDF.filter(sqft > 25000)

C. storesDF.where(storesDF[sqft] > 25000)

D. storesDF.where(sqft > 25000)

E. storesDF.filter(col("sqft") <= 25000)

Question #21*Topic 1*

Which of the following code blocks returns a DataFrame containing only the rows from DataFrame storesDF where the value in column sqft is less than or equal to 25,000 OR the value in column customerSatisfaction is greater than or equal to 30?

A. storesDF.filter(col("sqft") <= 25000 | col("customerSatisfaction") >= 30)

B. storesDF.filter(col("sqft") <= 25000 or col("customerSatisfaction") >= 30)

C. storesDF.filter(sqft <= 25000 or customerSatisfaction >= 30)

D. storesDF.filter(col(sqft) <= 25000 | col(customerSatisfaction) >= 30)

E. storesDF.filter((col("sqft") <= 25000) | (col("customerSatisfaction") >= 30))

Question #22*Topic 1*

Which of the following code blocks returns a new DataFrame from DataFrame storesDF where column storeId is of the type string?

A. storesDF.withColumn("storeId, cast(col("storeId"), StringType()))

B. storesDF.withColumn("storeId, col("storeId").cast(StringType()))

C. storesDF.withColumn("storeId, cast(storeId).as(StringType)

D. storesDF.withColumn("storeId, col(storeId).cast(StringType)

E. storesDF.withColumn("storeId, cast("storeId").as(StringType()))

Question #23*Topic 1*

Which of the following code blocks returns a new DataFrame with a new column employeesPerSqft that is the quotient of column numberOfEmployees and column sqft, both of which are from DataFrame storesDF? Note that column employeesPerSqft is not in the original DataFrame storesDF.

A. storesDF.withColumn("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

B. storesDF.withColumn("employeesPerSqft", "numberOfEmployees" / "sqft")

C. storesDF.select("employeesPerSqft", "numberOfEmployees" / "sqft")

D. storesDF.select("employeesPerSqft", col("numberOfEmployees") / col("sqft"))

E. storesDF.withColumn(col("employeesPerSqft"), col("numberOfEmployees") / col("sqft"))

Question #24*Topic 1*

The code block shown below should return a new DataFrame from DataFrame storesDF where column modality is the constant string "PHYSICAL", Assume DataFrame storesDF is the only defined language variable. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.  
Code block:  
storesDF. \_1\_(\_2\_,\_3\_(\_4\_))

A. 1. withColumn  
2. "modality"  
3. col  
4. "PHYSICAL"

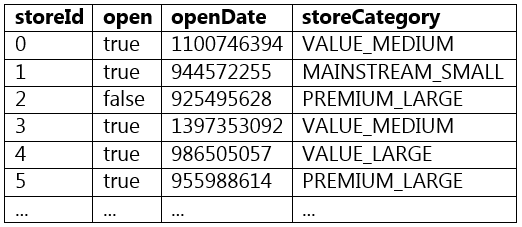
B. 1. withColumn  
2. "modality"  
3. lit  
4. PHYSICAL

C. 1. withColumn  
2. "modality"  
3. lit  
4. "PHYSICAL"

D. 1. withColumn  
2. "modality"  
3. SrtringType  
4. "PHYSICAL"

E. 1. newColumn  
2. modality  
3. SrtringType  
4. PHYSICAL

Question #25*Topic 1*

Which of the following code blocks returns a DataFrame where column storeCategory from DataFrame storesDF is split at the underscore character into column storeValueCategory and column storeSizeCategory?  
A sample of DataFrame storesDF is displayed below:  


A. (storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "\_")[1])  
.withColumn("storeSizeCategory", split(col("storeCategory"), "\_")[2]))

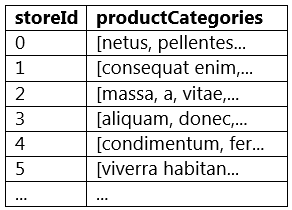
B. (storesDF.withColumn("storeValueCategory", col("storeCategory").split("\_")[0])  
.withColumn("storeSizeCategory", col("storeCategory").split("\_")[1]))

C. (storesDF.withColumn("storeValueCategory", split(col("storeCategory"), "\_")[0])  
.withColumn("storeSizeCategory", split(col("storeCategory"), "\_")[1]))

D. (storesDF.withColumn("storeValueCategory", split("storeCategory", "\_")[0])  
.withColumn("storeSizeCategory", split("storeCategory", "\_")[1]))

E. (storesDF.withColumn("storeValueCategory", col("storeCategory").split("\_")[1])  
.withColumn("storeSizeCategory", col("storeCategory").split("\_")[2]))

Question #26*Topic 1*

Which of the following code blocks returns a new DataFrame where column productCategories only has one word per row, resulting in a DataFrame with many more rows than DataFrame storesDF?  
A sample of storesDF is displayed below:  


A. storesDF.withColumn("productCategories", explode(col("productCategories")))

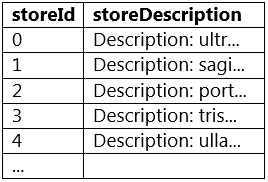
B. storesDF.withColumn("productCategories", split(col("productCategories")))

C. storesDF.withColumn("productCategories", col("productCategories").explode())

D. storesDF.withColumn("productCategories", col("productCategories").split())

E. storesDF.withColumn("productCategories", explode("productCategories"))

Question #27*Topic 1*

Which of the following code blocks returns a new DataFrame with column storeDescription where the pattern "Description: " has been removed from the beginning of column storeDescription in DataFrame storesDF?  
A sample of DataFrame storesDF is below:  


A. storesDF.withColumn("storeDescription", regexp\_replace(col("storeDescription"), "^Description: "))

B. storesDF.withColumn("storeDescription", col("storeDescription").regexp\_replace("^Description: ", ""))

C. storesDF.withColumn("storeDescription", regexp\_extract(col("storeDescription"), "^Description: ", ""))

D. storesDF.withColumn("storeDescription", regexp\_replace("storeDescription", "^Description: ", ""))

E. storesDF.withColumn("storeDescription", regexp\_replace(col("storeDescription"), "^Description: ", ""))

Question #28*Topic 1*

Which of the following code blocks returns a new DataFrame where column division from DataFrame storesDF has been replaced and renamed to column state and column managerName from DataFrame storesDF has been replaced and renamed to column managerFullName?

A. (storesDF.withColumnRenamed(["division", "state"], ["managerName", "managerFullName"])

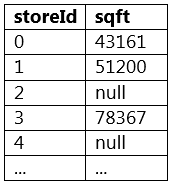
B. (storesDF.withColumn("state", col("division"))  
.withColumn("managerFullName", col("managerName")))

C. (storesDF.withColumn("state", "division")  
.withColumn("managerFullName", "managerName"))

D. (storesDF.withColumnRenamed("state", "division")  
.withColumnRenamed("managerFullName", "managerName"))

E. (storesDF.withColumnRenamed("division", "state")  
.withColumnRenamed("managerName", "managerFullName"))

Question #29*Topic 1*

The code block shown contains an error. The code block is intended to return a new DataFrame where column sqft from DataFrame storesDF has had its missing values replaced with the value 30,000. Identify the error.  
A sample of DataFrame storesDF is displayed below:  
  
Code block:  
storesDF.na.fill(30000, col("sqft"))

A. The argument to the subset parameter of fill() should be a string column name or a list of string column names rather than a Column object.

B. The na.fill() operation does not work and should be replaced by the dropna() operation.

C. he argument to the subset parameter of fill() should be a the numerical position of the column rather than a Column object.

D. The na.fill() operation does not work and should be replaced by the nafill() operation.

E. The na.fill() operation does not work and should be replaced by the fillna() operation.

Question #30*Topic 1*

Which of the following operations fails to return a DataFrame with no duplicate rows?

A. DataFrame.dropDuplicates()

B. DataFrame.distinct()

C. DataFrame.drop\_duplicates()

D. DataFrame.drop\_duplicates(subset = None)

E. DataFrame.drop\_duplicates(subset = "all")

Question #31*Topic 1*

Which of the following code blocks will most quickly return an approximation for the number of distinct values in column division in DataFrame storesDF?

A. storesDF.agg(approx\_count\_distinct(col("division")).alias("divisionDistinct"))

B. storesDF.agg(approx\_count\_distinct(col("division"), 0.01).alias("divisionDistinct"))

C. storesDF.agg(approx\_count\_distinct(col("division"), 0.15).alias("divisionDistinct"))

D. storesDF.agg(approx\_count\_distinct(col("division"), 0.0).alias("divisionDistinct"))

E. storesDF.agg(approx\_count\_distinct(col("division"), 0.05).alias("divisionDistinct"))

Question #32*Topic 1*

The code block shown below contains an error. The code block is intended to return a new DataFrame with the mean of column sqft from DataFrame storesDF in column sqftMean. Identify the error.  
Code block:  
storesDF.agg(mean("sqft").alias("sqftMean"))

A. The argument to the mean() operation should be a Column abject rather than a string column name.

B. The argument to the mean() operation should not be quoted.

C. The mean() operation is not a standalone function – it’s a method of the Column object.

D. The agg() operation is not appropriate here – the withColumn() operation should be used instead.

E. The only way to compute a mean of a column is with the mean() method from a DataFrame.

Question #33*Topic 1*

Which of the following operations can be used to return the number of rows in a DataFrame?

A. DataFrame.numberOfRows()

B. DataFrame.n()

C. DataFrame.sum()

D. DataFrame.count()

E. DataFrame.countDistinct()

Question #34*Topic 1*

Which of the following operations returns a GroupedData object?

A. DataFrame.GroupBy()

B. DataFrame.cubed()

C. DataFrame.group()

D. DataFrame.groupBy()

E. DataFrame.grouping\_id()

Question #35*Topic 1*

Which of the following code blocks returns a collection of summary statistics for all columns in  
DataFrame storesDF?

A. storesDF.summary("mean")

B. storesDF.describe(all = True)

C. storesDF.describe("all")

D. storesDF.summary("all")

E. storesDF.describe()

Question #36*Topic 1*

Which of the following code blocks fails to return a DataFrame reverse sorted alphabetically based on column division?

A. storesDF.orderBy("division", ascending – False)

B. storesDF.orderBy(["division"], ascending = [0])

C. storesDF.orderBy(col("division").asc())

D. storesDF.sort("division", ascending – False)

E. storesDF.sort(desc("division"))

Question #37*Topic 1*

Which of the following code blocks returns a 15 percent sample of rows from DataFrame storesDF without replacement?

A. storesDF.sample(fraction = 0.10)

B. storesDF.sampleBy(fraction = 0.15)

C. storesDF.sample(True, fraction = 0.10)

D. storesDF.sample()

E. storesDF.sample(fraction = 0.15)

Question #38*Topic 1*

Which of the following code blocks returns all the rows from DataFrame storesDF?

A. storesDF.head()

B. storesDF.collect()

C. storesDF.count()

D. storesDF.take()

E. storesDF.show()

Question #39*Topic 1*

Which of the following code blocks applies the function assessPerformance() to each row of DataFrame storesDF?

A. [assessPerformance(row) for row in storesDF.take(3)]

B. [assessPerformance() for row in storesDF]

C. storesDF.collect().apply(lambda: assessPerformance)

D. [assessPerformance(row) for row in storesDF.collect()]

E. [assessPerformance(row) for row in storesDF]

Question #40*Topic 1*

The code block shown below contains an error. The code block is intended to print the schema of DataFrame storesDF. Identify the error.  
Code block:  
storesDF.printSchema

A. There is no printSchema member of DataFrame – schema and the print() function should be used instead.

B. The entire line needs to be a string – it should be wrapped by str().

C. There is no printSchema member of DataFrame – the getSchema() operation should be used instead.

D. There is no printSchema member of DataFrame – the schema() operation should be used instead.

E. The printSchema member of DataFrame is an operation and needs to be followed by parentheses.

Question #41*Topic 1*

The code block shown below should create and register a SQL UDF named "ASSESS\_PERFORMANCE" using the Python function assessPerformance() and apply it to column customerSatisfaction in table stores. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.  
Code block:  
spark.\_1\_.\_2\_(\_3\_, \_4\_)  
spark.sql("SELECT customerSatisfaction, \_5\_(customerSatisfaction) AS result FROM stores")

A. 1. udf  
2. register  
3. "ASSESS\_PERFORMANCE"  
4. assessPerformance  
5. ASSESS\_PERFORMANCE

B. 1. udf  
2. register  
3. assessPerformance  
4. "ASSESS\_PERFORMANCE"  
5. "ASSESS\_PERFORMANCE"

C. 1. udf  
2. register  
3."ASSESS\_PERFORMANCE"  
4. assessPerformance  
5. "ASSESS\_PERFORMANCE"

D. 1. register  
2. udf  
3. "ASSESS\_PERFORMANCE"  
4. assessPerformance  
5. "ASSESS\_PERFORMANCE"

E. 1. udf  
2. register  
3. ASSESS\_PERFORMANCE  
4. assessPerformance  
5. ASSESS\_PERFORMANCE

Question #42*Topic 1*

The code block shown below contains an error. The code block is intended to create a Python UDF assessPerformanceUDF() using the integer-returning Python function assessPerformance() and apply it to column customerSatisfaction in DataFrame storesDF. Identify the error.  
Code block:  
assessPerformanceUDF – udf(assessPerformance)  
storesDF.withColumn("result", assessPerformanceUDF(col("customerSatisfaction")))

A. The assessPerformance() operation is not properly registered as a UDF.

B. The withColumn() operation is not appropriate here – UDFs should be applied by iterating over rows instead.

C. UDFs can only be applied vie SQL and not through the DataFrame API.

D. The return type of the assessPerformanceUDF() is not specified in the udf() operation.

E. The assessPerformance() operation should be used on column customerSatisfaction rather than the assessPerformanceUDF() operation.

Question #43*Topic 1*

The code block shown below contains an error. The code block is intended to use SQL to return a new DataFrame containing column storeId and column managerName from a table created from DataFrame storesDF. Identify the error.  
Code block:  
storesDF.createOrReplaceTempView("stores")  
storesDF.sql("SELECT storeId, managerName FROM stores")

A. The createOrReplaceTempView() operation does not make a Dataframe accessible via SQL.

B. The sql() operation should be accessed via the spark variable rather than DataFrame storesDF.

C. There is the sql() operation in DataFrame storesDF. The operation query() should be used instead.

D. This cannot be accomplished using SQL – the DataFrame API should be used instead.

E. The createOrReplaceTempView() operation should be accessed via the spark variable rather than DataFrame storesDF.

Question #44*Topic 1*

The code block shown below should create a single-column DataFrame from Python list years which is made up of integers. Choose the response that correctly fills in the numbered blanks within the code block to complete this task.  
Code block:  
\_1\_.\_2\_(\_3\_, \_4\_)

A. 1. spark  
2. createDataFrame  
3. years  
4. IntegerType

B. 1. DataFrame  
2. create  
3. [years]  
4. IntegerType

C. 1. spark  
2. createDataFrame  
3. [years]  
4. IntegertType

D. 1. spark  
2. createDataFrame  
3. [years]  
4. IntegertType()

E. 1. spark  
2. createDataFrame  
3. years  
4. IntegertType()