DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Q&As

Databricks Certified Professional Data Engineer Exam

Pass Databricks DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.passapply.com/databricks-certified-professional-data-engineer.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Databricks Official Exam Center

QUESTION 1

Review the following error traceback:

```
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
<command-3293767849433948> in <module>
----> 1 display(df.select(3*"heartrate"))

/databricks/spark/python/pyspark/sql/dataframe.py in select(self, *cols)
   1690         [Row(name='Alice', age=12), Row(name='Bob', age=15)]
   1691         """
-> 1692         jdf = self._jdf.select(self._jcols(*cols))
   1693         return DataFrame(jdf, self.sql_ctx)
   1694

/databricks/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in __call__(self, *args)
   1302
   1303         answer = self.gateway_client.send_command(command)
-> 1304         return_value = get_return_value(
   1305             answer, self.gateway_client, self.target_id, self.name)
   1306

/databricks/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
    121                 # Hide where the exception came from that shows a non-Pythonic
    122                 # JVM exception message.
--> 123                 raise converted from None
    124             else:
    125                 raise

AnalysisException: cannot resolve '`heartrateheartrateheartrate`' given input columns:
[spark_catalog.database.table.device_id, spark_catalog.database.table.heartrate,
spark_catalog.database.table.mrn, spark_catalog.database.table.time];
'Project ['heartrateheartrateheartrate]
+- SubqueryAlias spark_catalog.database.table
   +- Relation[device_id#75L,heartrate#76,mrn#77L,time#78] parquet
```

Which statement describes the error being raised?

A. The code executed was PvSoark but was executed in a Scala notebook.

B. There is no column in the table named heartrateheartrateheartrate

C. There is a type error because a column object cannot be multiplied.

D. There is a type error because a DataFrame object cannot be multiplied.

E. There is a syntax error because the heartrate column is not correctly identified as a column.

Correct Answer: E

Explanation: The error is a Py4JJavaError, which means that an exception was thrown in Java code called by Python code using Py4J. Py4J is a library that enables Python programs to dynamically access Java objects in a Java Virtual

Machine (JVM). PySpark uses Py4J to communicate with Spark\\'s JVM-based engine. The error message shows that

the exception was thrown by org.apache.spark.sql.AnalysisException, which means that an error occurred during the

analysis phase of Spark SQL query processing. The error message also shows that the cause of the exception was "cannot resolve `heartrateheartrateheartrate\\' given input columns". This means that Spark could not find a column named

heartrateheartrateheartrate in the input DataFrame or Dataset. The reason for this error is that there is a syntax error in the code that caused this exception. The code

is:

df.withColumn("heartrate", heartrate * 3)

The code tries to create a new column called heartrate by multiplying an existing column called heartrate by 3. However, the code does not correctly identify the heartrate column as a column object, but rather as a plain Python variable. This

causes PySpark to concatenate the variable name with itself three times, resulting in heartrateheartrateheartrate, which is not a valid column name. To fix this error, the code should use one of the following ways to identify the heartrate

column as a column object:

df.withColumn("heartrate", df["heartrate"] * 3) df.withColumn("heartrate", df.heartrate * 3) df.withColumn("heartrate", col("heartrate") * 3)

Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Py4J Documentation, under "What is Py4J?"; Databricks Documentation, under "Query plans - Analysis phase"; Databricks Documentation,

under "Accessing columns".

QUESTION 2

The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.

What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?

A. Can Manage

B. Can Edit

C. No permissions

D. Can Read

E. Can Run

Correct Answer: D

Explanation: This is the correct answer because it is the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data. Notebook permissions are used to control access to notebooks in Databricks workspaces. There are four types of notebook permissions: Can Manage, Can Edit, Can Run, and Can Read. Can Manage allows full control over the notebook, including editing, running, deleting, exporting,

and changing permissions. Can Edit allows modifying and running the notebook, but not changing permissions or deleting it. Can Run allows executing commands in an existing cluster attached to the notebook, but not modifying or exporting it. Can Read allows viewing the notebook content, but not running or modifying it. In this case, granting Can Read permission to the user will allow them to review the production logic in the notebook without allowing them to makeany changes to it or run any commands that may affect production data. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Notebook permissions" section.

QUESTION 3

Which statement characterizes the general programming model used by Spark Structured Streaming?

A. Structured Streaming leverages the parallel processing of GPUs to achieve highly parallel data throughput.

B. Structured Streaming is implemented as a messaging bus and is derived from Apache Kafka.

C. Structured Streaming uses specialized hardware and I/O streams to achieve sub- second latency for data transfer.

D. Structured Streaming models new data arriving in a data stream as new rows appended to an unbounded table.

E. Structured Streaming relies on a distributed network of nodes that hold incremental state values for cached stages.

Correct Answer: D

Explanation: This is the correct answer because it characterizes the general programming model used by Spark Structured Streaming, which is to treat a live data stream as a table that is being continuously appended. This leads to a new stream processing model that is very similar to a batch processing model, where users can express their streaming computation using the same Dataset/DataFrame API as they would use for static data. The Spark SQL engine will take care of running the streaming query incrementally and continuously and updating the final result as streaming data continues to arrive. Verified References: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Overview" section.

QUESTION 4

Each configuration below is identical to the extent that each cluster has 400 GB total of RAM, 160 total cores and only one Executor per VM.

Given a job with at least one wide transformation, which of the following cluster configurations will result in maximum performance?

A. Total VMs; 1 400 GB per Executor 160 Cores / Executor

B. Total VMs: 8 50 GB per Executor 20 Cores / Executor

C. Total VMs: 4 100 GB per Executor 40 Cores/Executor

D. Total VMs:2 200 GB per Executor 80 Cores / Executor

Correct Answer: B

Explanation: This is the correct answer because it is the cluster configuration that will result in maximum performance for a job with at least one wide transformation. A wide transformation is a type of transformation that requires shuffling data across partitions, such as join, groupBy, or orderBy. Shuffling can be expensive and time-consuming, especially if

there are too many or too few partitions. Therefore, it is important to choose a cluster configuration that can balance the trade-off between parallelism and network overhead. In this case, having 8 VMs with 50 GB per executor and 20 cores per executor will create 8 partitions, each with enough memory and CPU resources to handle the shuffling efficiently. Having fewer VMs with more memory and cores per executor will create fewer partitions, which will reduce parallelism and increase the size of each shuffle block. Having more VMs with less memory and cores per executor will create more partitions, which will increase parallelism but also increase the network overhead and the number of shuffle files. Verified References: [Databricks Certified Data Engineer Professional], under "Performance Tuning" section; Databricks Documentation, under "Cluster configurations" section.

QUESTION 5

A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. At present, during normal execution, each microbatch of data is processed in less than 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a trigger interval of 10 seconds.

Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

A. Decrease the trigger interval to 5 seconds; triggering batches more frequently allows idle executors to begin processing the next batch while longer running tasks from previous batches finish.

B. Increase the trigger interval to 30 seconds; setting the trigger interval near the maximum execution time observed for each batch is always best practice to ensure no records are dropped.

C. The trigger interval cannot be modified without modifying the checkpoint directory; to maintain the current stream state, increase the number of shuffle partitions to maximize parallelism.

D. Use the trigger once option and configure a Databricks job to execute the query every 10 seconds; this ensures all backlogged records are processed with each batch.

E. Decrease the trigger interval to 5 seconds; triggering batches more frequently may prevent records from backing up and large batches from causing spill.

Correct Answer: D

Explanation: This is the correct answer because it can meet the requirement of processing records in less than 10 seconds without modifying the checkpoint directory or dropping records. The trigger once option is a special type of trigger that runs the streaming query only once and terminates after processing all available data. This option can be useful for scenarios where you want to run streaming queries on demand or periodically, rather than continuously. By using the trigger once option and configuring a Databricks job to execute the query every 10 seconds, you can ensure that all backlogged records are processed with each batch and avoid inconsistent execution times. Verified References: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Trigger Once" section.