DATABRICKS-CERTIFIED-PR OFESSIONAL-DATA-ENGINEER$^{Q\&As}$

Databricks Certified Professional Data Engineer Exam

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

SATISFACTION GUARANTEED
100%
SATISFACTION GUARANTEED

QUESTION 1

Which statement describes integration testing?

A. Validates interactions between subsystems of your application

B. Requires an automated testing framework

C. Requires manual intervention

D. Validates an application use case

E. Validates behavior of individual elements of your application

Correct Answer: A

Explanation: This is the correct answer because it describes integration testing. Integration testing is a type of testing that validates interactions between subsystems of your application, such as modules, components, or services. Integration testing ensures that the subsystems work together as expected and produce the correct outputs or results. Integration testing can be done at different levels of granularity, such as component integration testing, system integration testing, or end-to-end testing. Integration testing can help detect errors or bugs that may not be found by unit testing, which only validates behavior of individual elements of your application. Verified References: [Databricks Certified Data Engineer Professional], under "Testing" section; Databricks Documentation, under "Integration testing" section.

QUESTION 2

A data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs. A DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens.

Which statement describes the contents of the workspace audit logs concerning these events?

A. Because the REST API was used for job creation and triggering runs, a Service Principal will be automatically used to identity these events.

B. Because User B last configured the jobs, their identity will be associated with both the job creation events and the job run events.

C. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events.

D. Because the REST API was used for job creation and triggering runs, user identity will not be captured in the audit logs.

E. Because User A created the jobs, their identity will be associated with both the job creation events and the job run events.

Correct Answer: C

Explanation: The events are that a data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs, and a DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens. The workspace audit logs are logs that record user activities in a Databricks workspace, such as creating, updating, or deleting objects like clusters, jobs, notebooks, or tables. The workspace audit logs also capture the identity of the user who performed each activity, as well as the time and details of the activity. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events in the workspace audit logs. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Workspace audit logs" section.

QUESTION 3

Which statement regarding stream-static joins and static Delta tables is correct?

A. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch.

B. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of the job\\'s initialization.

C. The checkpoint directory will be used to track state information for the unique keys present in the join.

D. Stream-static joins cannot use static Delta tables because of consistency issues.

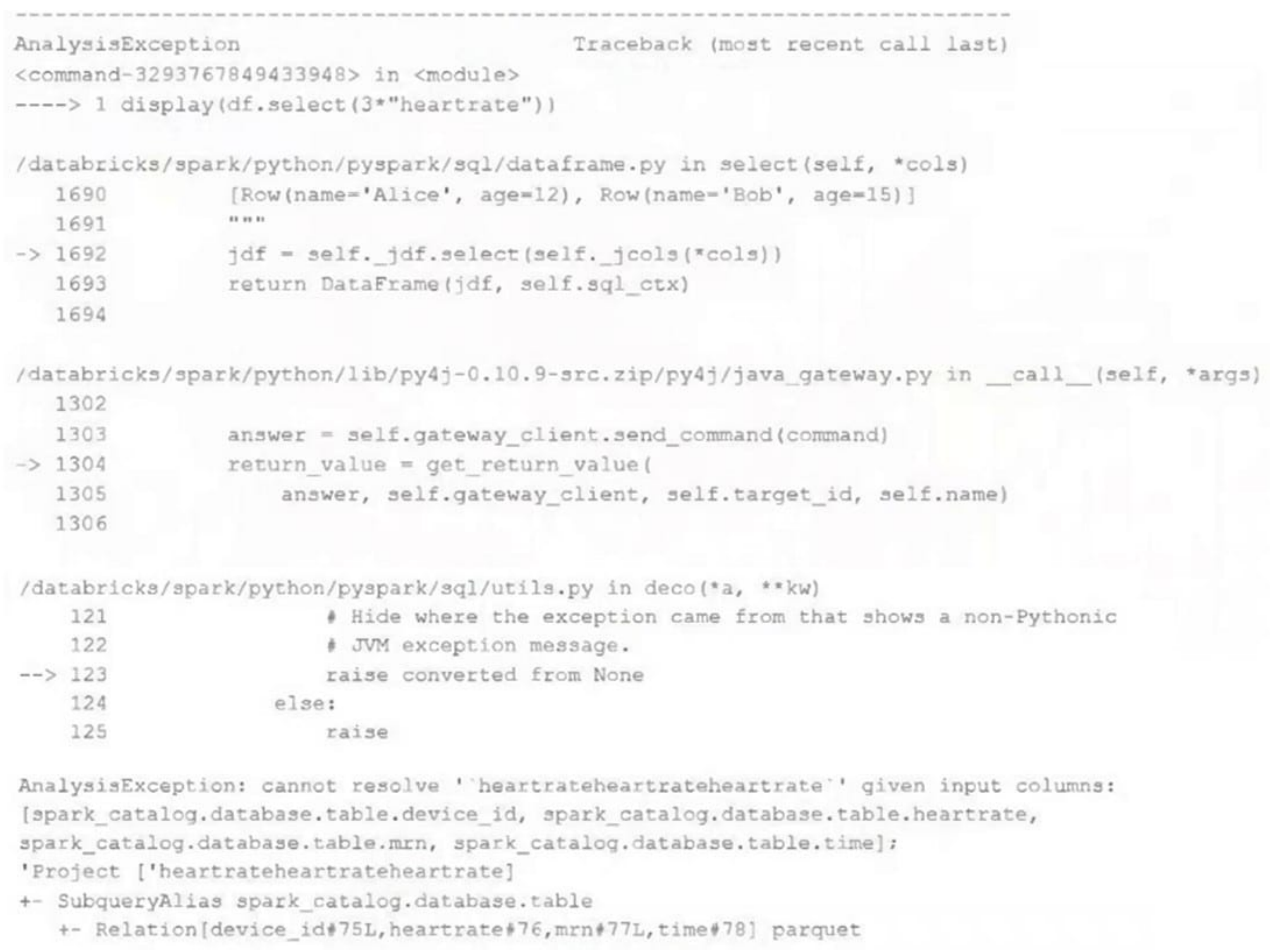
E. The checkpoint directory will be used to track updates to the static Delta table.

Correct Answer: A

Explanation: This is the correct answer because stream-static joins are supported by Structured Streaming when one of the tables is a static Delta table. A static Delta table is a Delta table that is not updated by any concurrent writes, such as appends or merges, during the execution of a streaming query. In this case, each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch, which means it will reflect any changes made to the static Delta table before the start of each microbatch. Verified References:[Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Stream and static joins" section.

QUESTION 4

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 5

The data engineering team has configured a job to process customer requests to be forgotten (have their data deleted). All user data that needs to be deleted is stored in Delta Lake tables using default table settings.

The team has decided to process all deletions from the previous week as a batch job at 1am each Sunday. The total duration of this job is less than one hour. Every Monday at 3am, a batch job executes a series ofVACUUMcommands on all Delta Lake tables throughout the organization.

The compliance officer has recently learned about Delta Lake\\'s time travel functionality. They are concerned that this might allow continued access to deleted data.

Assuming all delete logic is correctly implemented, which statement correctly addresses this concern?

A. Because the vacuum command permanently deletes all files containing deleted records, deleted records may be accessible with time travel for around 24 hours.

B. Because the default data retention threshold is 24 hours, data files containing deleted records will be retained until the vacuum job is run the following day.

C. Because Delta Lake time travel provides full access to the entire history of a table, deleted records can always be recreated by users with full admin privileges.

D. Because Delta Lake\\'s delete statements have ACID guarantees, deleted records will be permanently purged from all storage systems as soon as a delete job completes.

E. Because the default data retention threshold is 7 days, data files containing deleted records will be retained until the vacuum job is run 8 days later.

Correct Answer: A

Explanation: This is the correct answer because Delta Lake\\'s delete statements do not physically remove the data files that contain the deleted records, but only mark them as logically deleted in the transaction log. These files are still

accessible with time travel until they are permanently deleted by the vacuum command. The default data retention threshold for vacuum is 7 days, but in this case it is overridden by setting it to 24 hours in each vacuum command. Therefore,

deleted records may be accessible with time travel for around 24 hours after they are deleted, until they are vacuumed. Verified References:

[Databricks Certified Data Engineer Professional], under "Delta Lake" section; [Databricks Documentation], under "Optimizations - Vacuum" section.