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

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

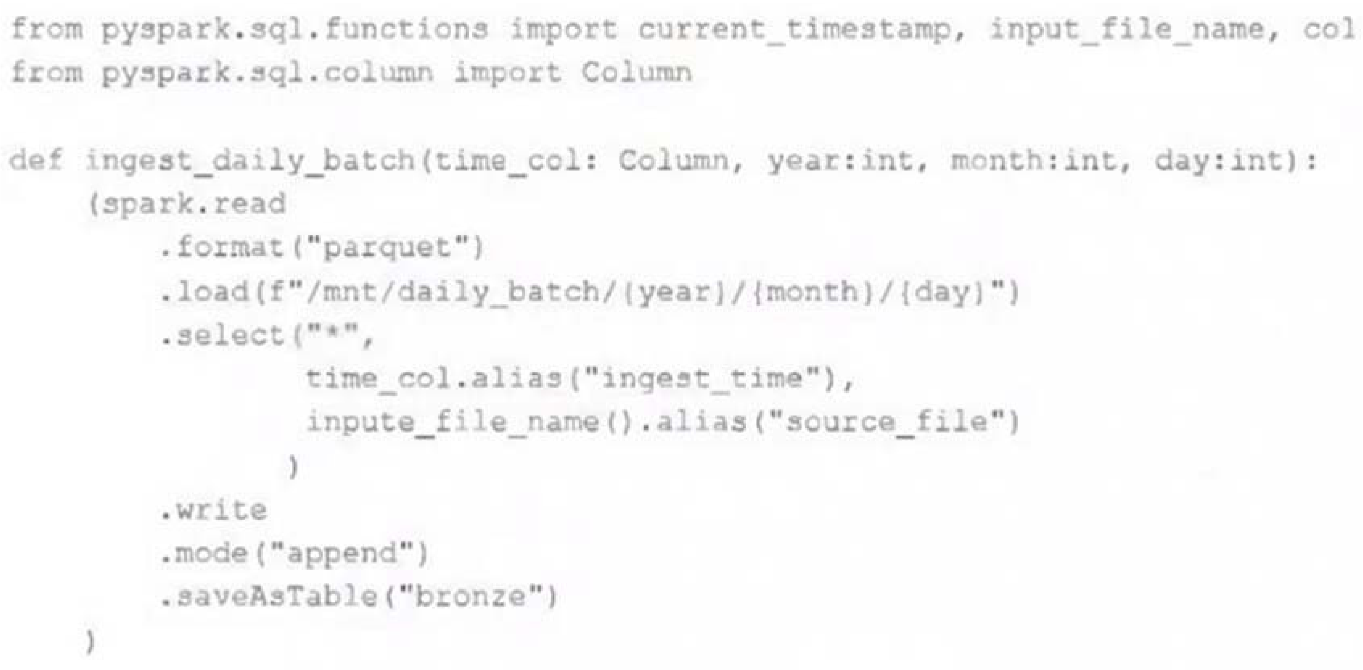
https://www.passapply.com/databricks-certified-professional-data-engineer.html

100% Passing Guarantee
100% Money Back Assurance

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

QUESTION 1

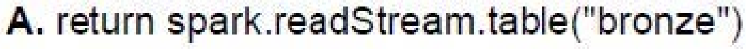
A nightly job ingests data into a Delta Lake table using the following code:



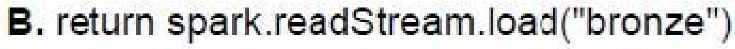
```
from pyspark.sql.functions import current_timestamp, input_file_name, col
from pyspark.sql.column import Column

def ingest_daily_batch(time_col: Column, year:int, month:int, day:int):
    (spark.read
        .format("parquet")
        .load(f"/mnt/daily_batch/{year}/{month}/{day}")
        .select("*",
                time_col.alias("ingest_time"),
                inpute_file_name().alias("source_file")
               )
        .write
        .mode("append")
        .saveAsTable("bronze")
    )
```

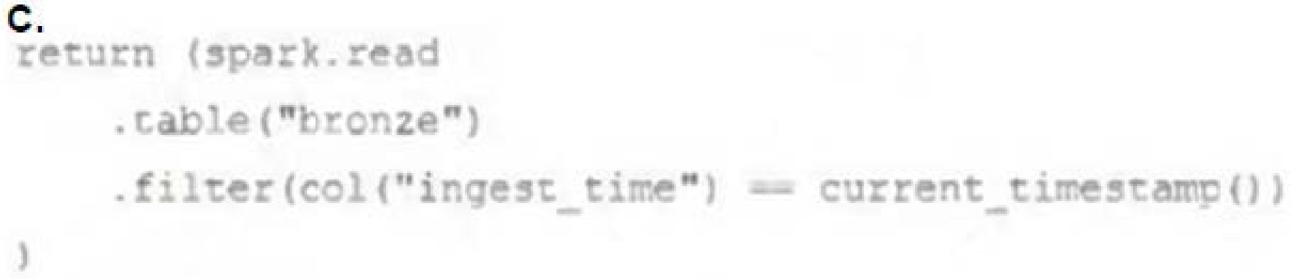
The next step in the pipeline requires a function that returns an object that can be used to manipulate new records that have not yet been processed to the next table in the pipeline. Which code snippet completes this function definition?



A. return spark.readStream.table("bronze")

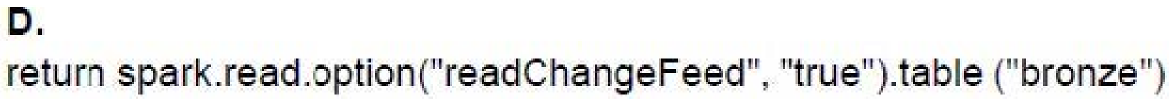


B. return spark.readStream.load("bronze")



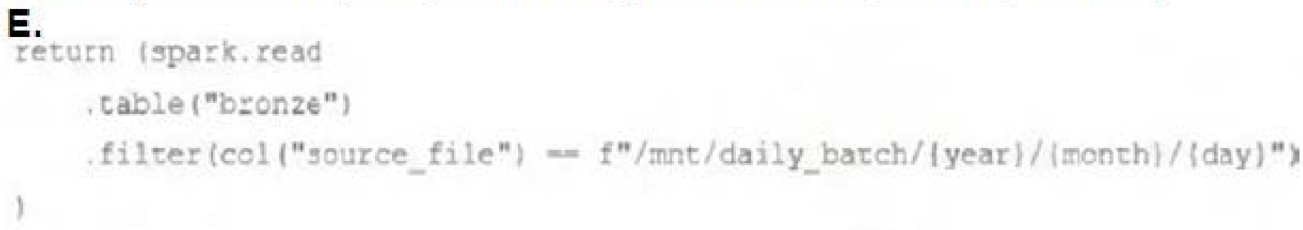
C.

```
return (spark.read
    .table("bronze")
    .filter(col("ingest_time") == current_timestamp())
)
```



D.

return spark.read.option("readChangeFeed", "true").table ("bronze")



E.

```
return (spark.read
    .table("bronze")
    .filter(col("source_file") == f"/mnt/daily_batch/{year}/{month}/{day}")
)
```

A. Option A

B. Option B

C. Option C

D. Option D

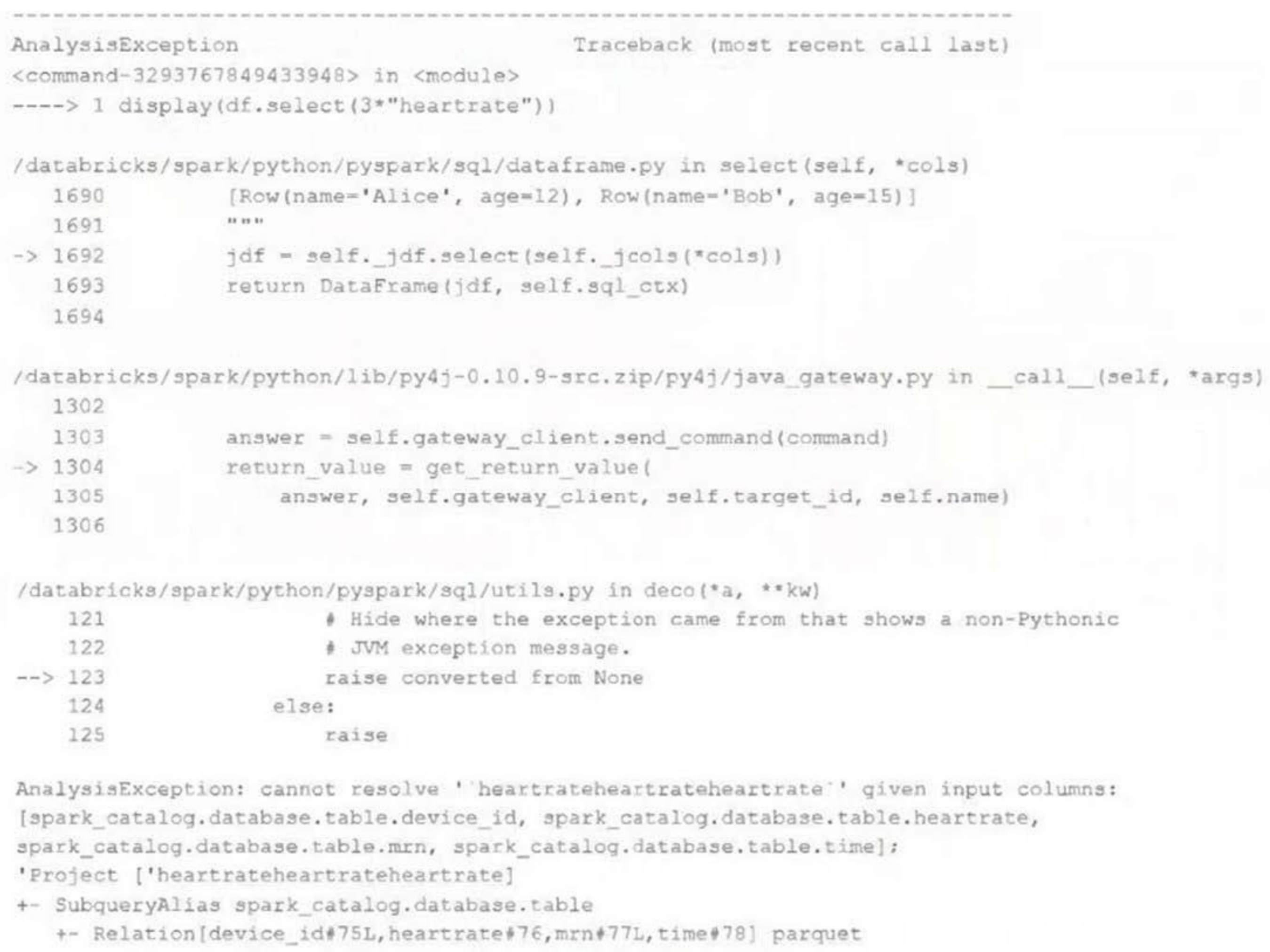
E. Option E

Correct Answer: E

https://docs.databricks.com/en/delta/delta-change-data-feed.html

QUESTION 2

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

The error being raised is an AnalysisException, which is a type of exception that occurs when Spark SQL cannot analyze or execute a query due to some logical or semantic error1. In this case, the error message indicates that the query cannot resolve the column name `heartrateheartrateheartrate\\' given the input columns `heartrate\\' and `age\\'. This means that there is no column in the table named `heartrateheartrateheartrate\\', and the query is invalid. A possible cause of this error is a typo or a copy-paste mistake in the query. To fix this error, the query should use a valid column name that exists in the table, such as `heartrate\\'. References: AnalysisException

QUESTION 3

Which statement describes Delta Lake optimized writes?

A. A shuffle occurs prior to writing to try to group data together resulting in fewer files instead of each executor writing multiple files based on directory partitions.

B. Optimized writes logical partitions instead of directory partitions partition boundaries are only represented in metadata fewer small files are written.

C. An asynchronous job runs after the write completes to detect if files could be further compacted; yes, an OPTIMIZE job is executed toward a default of 1 GB.

D. Before a job cluster terminates, OPTIMIZE is executed on all tables modified during the most recent job.

Correct Answer: A

Delta Lake optimized writes involve a shuffle operation before writing out data to the Delta table. The shuffle operation groups data by partition keys, which can lead to a reduction in the number of output files and potentially larger files, instead

of multiple smaller files. This approach can significantly reduce the total number of files in the table, improve read performance by reducing the metadata overhead, and optimize the table storage layout, especially for workloads with many

small files.

References:

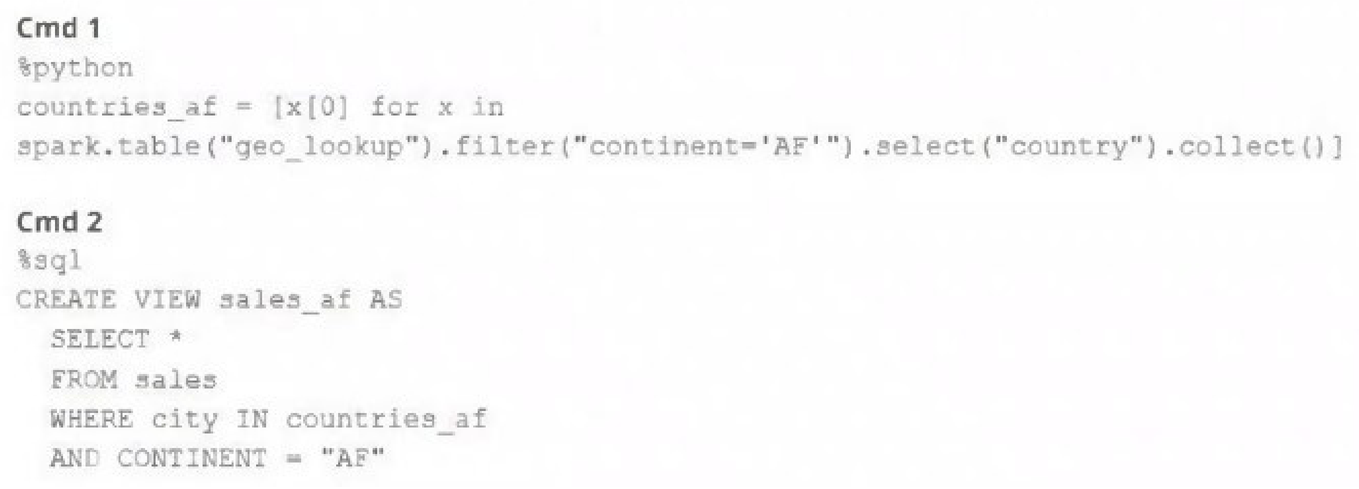
Databricks documentation on Delta Lake performance tuning:

https://docs.databricks.com/delta/optimizations/auto-optimize.html

QUESTION 4

A junior member of the data engineering team is exploring the language interoperability of Databricks notebooks. The intended outcome of the below code is to register a view of all sales that occurred in countries on the continent of Africa that appear in the geo_lookup table.

Before executing the code, running SHOW TABLES on the current database indicates the database contains only two tables: geo_lookup and sales.



Cmd 1

```
%python
countries_af = [x[0] for x in
spark.table("geo_lookup").filter("continent='AF'").select("country").collect()]
```

Cmd 2

```
%sql
CREATE VIEW sales_af AS
  SELECT *
  FROM sales
  WHERE city IN countries_af
  AND CONTINENT = "AF"
```

Which statement correctly describes the outcome of executing these command cells in order in an interactive notebook?

A. Both commands will succeed. Executing show tables will show that countries at and sales at have been registered as views.

B. Cmd 1 will succeed. Cmd 2 will search all accessible databases for a table or view named countries af: if this entity exists, Cmd 2 will succeed.

C. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable representing a PySpark DataFrame.

D. Both commands will fail. No new variables, tables, or views will be created.

E. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable containing a list of strings.

Correct Answer: E

This is the correct answer because Cmd 1 is written in Python and uses a list comprehension to extract the country names from the geo_lookup table and store them in a Python variable named countries af. This variable will contain a list of

strings, not a PySpark DataFrame or a SQL view. Cmd 2 is written in SQL and tries to create a view named sales af by selecting from the sales table where city is in countries af. However, this command will fail because countries af is not a

valid SQL entity and cannot be used in a SQL query. To fix this, a better approach would be to use spark.sql() to execute a SQL query in Python and pass the countries af variable as a parameter. Verified References:

[Databricks Certified Data Engineer Professional], under "Language Interoperability" section; Databricks Documentation, under "Mix languages" section.

QUESTION 5

A developer has successfully configured credential for Databricks Repos and cloned a remote Git repository. Hey don not have privileges to make changes to the main branch, which is the only branch currently visible in their workspace.

Use Response to pull changes from the remote Git repository commit and push changes to a branch that appeared as a changes were pulled.

A. Use Repos to merge all differences and make a pull request back to the remote repository.

B. Use repos to merge all difference and make a pull request back to the remote repository.

C. Use Repos to create a new branch commit all changes and push changes to the remote Git repertory.

D. Use repos to create a fork of the remote repository commit all changes and make a pull request on the source repository

Correct Answer: C

In Databricks Repos, when a user does not have privileges to make changes directly to the main branch of a cloned remote Git repository, the recommended approach is to create a new branch within the Databricks workspace. The developer can then make changes in this new branch, commit those changes, and push the new branch to the remote Git repository. This workflow allows for isolated development without affecting the main branch, enabling the developer to propose changes via a pull request from the new branch to the main branch in the remote repository. This method adheres to common Git collaboration workflows, fostering code review and collaboration while ensuring the integrity of the main branch. References: Databricks documentation on using Repos with Git: https://docs.databricks.com/repos.html