# 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.pass2lead.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 junior data engineer on your team has implemented the following code block.

```
MERGE INTO events
USING new_events
ON events.event_id = new_events.event_id
WHEN NOT MATCHED
  INSERT *
```

The viewnew_eventscontains a batch of records with the same schema as theeventsDelta table. Theevent_idfield serves as a unique key for this table. When this query is executed, what will happen with new records that have the sameevent_idas an existing record?

A. They are merged.

B. They are ignored.

C. They are updated.

D. They are inserted.

E. They are deleted.

Correct Answer: B

Explanation: This is the correct answer because it describes what will happen with new records that have the same event_id as an existing record when the query is executed. The query uses the INSERT INTO command to append new records from the view new_events to the table events. However, the INSERT INTO command does not check for duplicate values in the primary key column (event_id) and does not perform any update or delete operations on existing records. Therefore, if there are new records that have the same event_id as an existing record, they will be ignored and not inserted into the table events. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Append data using INSERT INTO" section.

---

**QUESTION 2**

An upstream system has been configured to pass the date for a given batch of data to the Databricks Jobs API as a parameter. The notebook to be scheduled will use this parameter to load data with the following code:

df = spark.read.format("parquet").load(f"/mnt/source/(date)")

Which code block should be used to create the date Python variable used in the above code block?

A. date = spark.conf.get("date")

B. input_dict = input() date= input_dict["date"]

C. import sys date = sys.argv[1]

D. date = dbutils.notebooks.getParam("date")

E. dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date")

Correct Answer: D

Explanation: This is the correct way to get a parameter passed to a notebook by the Databricks Jobs API. The dbutils.notebooks.getParam method returns the value of a parameter passed to a notebook as a string. If no parameter with that name is passed, it returns None by default. You can also specify a default value as a second argument. Verified References: Databricks Certified Data Engineer Professional, under "Databricks Tooling" section; Databricks Documentation, under "Pass parameters to a notebook" section.

**QUESTION 3**

A junior data engineer is working to implement logic for a Lakehouse table namedsilver_device_recordings. The source data contains 100 unique fields in a highly nested JSON structure.

Thesilver_device_recordingstable will be used downstream to power several production monitoring dashboards and a production model. At present, 45 of the 100 fields are being used in at least one of these applications.

The data engineer is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields.

Which of the following accurately presents information about Delta Lake and Databricks that may impact their decision-making process?

A. The Tungsten encodingused by Databricksis optimized for storing stringdata; newly- added nativesupport for querying JSON strings means that stringtypes are always most efficient.

B. Because Delta Lake uses Parquet for data storage, data types can be easily evolved by just modifying file footer information in place.

C. Human labor in writingcode is the largest cost associated with data engineering workloads; as such, automatingtable declaration logic should be a priority in all migration workloads.

D. Because Databricks will infer schema using types that allow all observed data to be processed, setting types manually provides greater assurance of data quality enforcement.

E. Schema inference and evolution on .Databricks ensure that inferred types will always accurately match the data types used by downstream systems.

Correct Answer: D

Explanation: This is the correct answer because it accurately presents information about Delta Lake and Databricks that may impact the decision-making process of a junior data engineer who is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields. Delta Lake and Databricks support schema inference and evolution, which means that they can automatically infer the schema of a table from the source data and allow adding new columns or changing column types without affecting existing queries or pipelines. However, schema inference and evolution may not always be desirable or reliable, especially when dealing with complex or nested data structures or when enforcing data quality and consistency across different systems. Therefore, setting types manually can provide greater assurance of data quality enforcement and avoid potential errors or conflicts due to incompatible or unexpected data types. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Schema inference and partition of streaming DataFrames/ Datasets" section.

**QUESTION 4**

Which distribution does Databricks support for installing custom Python code packages?

A. sbt

B. CRAN

C. CRAM

D. nom

E. Wheels

F. jars

Correct Answer: D

**QUESTION 5**

A new data engineer notices that a critical field was omitted from an application that writes its Kafka source to Delta Lake. This happened even though the critical field was in the Kafka source. That field was further missing from data written to dependent, long-term storage. The retention threshold on the Kafka service is seven days. The pipeline has been in production for three months.

Which describes how Delta Lake can help to avoid data loss of this nature in the future?

A. The Delta log and Structured Streaming checkpoints record the full history of the Kafka producer.

B. Delta Lake schema evolution can retroactively calculate the correct value for newly added fields, as long as the data was in the original source.

C. Delta Lake automatically checks that all fields present in the source data are included in the ingestion layer.

D. Data can never be permanently dropped or deleted from Delta Lake, so data loss is not possible under any circumstance.

E. Ingestine all raw data and metadata from Kafka to a bronze Delta table creates a permanent, replayable history of the data state.

Correct Answer: E

Explanation: This is the correct answer because it describes how Delta Lake can help to avoid data loss of this nature in the future. By ingesting all raw data and metadata from Kafka to a bronze Delta table, Delta Lake creates a permanent, replayable history of the data state that can be used for recovery or reprocessing in case of errors or omissions in downstream applications or pipelines. Delta Lake also supports schema evolution, which allows adding new columns to existing tables without affecting existing queries or pipelines. Therefore, if a critical field was omitted from an application that writes its Kafka source to Delta Lake, it can be easily added later and the data can be reprocessed from the bronze table without losing any information. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Delta Lake core features" section.

**QUESTION 6**

The downstream consumers of a Delta Lake table have been complaining about data quality issues impacting performance in their applications. Specifically, they have complained that invalidlatitudeandlongitudevalues in theactivity_detailstable have been breaking their ability to use other geolocation processes.

A junior engineer has written the following code to addCHECKconstraints to the Delta Lake table: A senior engineer has confirmed the above logic is correct and the valid ranges for latitude and longitude are provided, but the code fails when executed. Which statement explains the cause of this failure?

```
ALTER TABLE activity_details
ADD CONSTRAINT valid_coordinates
CHECK (
  latitude >= -90 AND
  latitude <= 90 AND
  longitude >= -180 AND
  longitude <= 180);
```

A. Because another team uses this table to support a frequently running application, two- phase locking is preventing the operation from committing.

B. The activity details table already exists; CHECK constraints can only be added during initial table creation.

C. The activity details table already contains records that violate the constraints; all existing data must pass CHECK constraints in order to add them to an existing table.

D. The activity details table already contains records; CHECK constraints can only be added prior to inserting values into a table.

E. The current table schema does not contain the field valid coordinates; schema evolution will need to be enabled before altering the table to add a constraint.

Correct Answer: C

Explanation: The failure is that the code to add CHECK constraints to the Delta Lake table fails when executed. The code uses ALTER TABLE ADD CONSTRAINT commands to add two CHECK constraints to a table named activity_details. The first constraint checks if the latitude value is between -90 and 90, and the second constraint checks if the longitude value is between -180 and 180. The cause of this failure is that the activity_details table already contains records that violate these constraints, meaning that they have invalid latitude or longitude values outside of these ranges. When adding CHECK constraints to an existing table, Delta Lake verifies that all existing data satisfies the constraints before adding them to the table. If any record violates the constraints, Delta Lake throws an exception and aborts the operation. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Add a CHECK constraint to an existing table" section.

**QUESTION 7**

Which Python variable contains a list of directories to be searched when trying to locate required modules?

A. importlib.resource path

B. ,sys.path

C. os-path

D. pypi.path

E. pylib.source

Correct Answer: B

**QUESTION 8**

Although the Databricks Utilities Secrets module provides tools to store sensitive credentials and avoid accidentally displaying them in plain text users should still be careful with which credentials are stored here and which users have access to using these secrets.

Which statement describes a limitation of Databricks Secrets?

A. Because the SHA256 hash is used to obfuscate stored secrets, reversing this hash will display the value in plain text.

B. Account administrators can see all secrets in plain text by loggingon to the Databricks Accounts console.

C. Secrets are stored in an administrators-only table within the Hive Metastore; database administrators have permission to query this table by default.

D. Iterating through a stored secret and printing each character will display secret contents in plain text.

E. The Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials.

Correct Answer: E

Explanation: This is the correct answer because it describes a limitation of Databricks Secrets. Databricks Secrets is a module that provides tools to store sensitive credentials and avoid accidentally displaying them in plain text. Databricks Secrets allows creating secret scopes, which are collections of secrets that can be accessed by users or groups. Databricks Secrets also allows creating and managing secrets using the Databricks CLI or the Databricks REST API. However, a limitation of Databricks Secrets is that the Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials. Therefore, users should still be careful with which credentials are stored in Databricks Secrets and which users have access to using these secrets. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "List secrets" section.

**QUESTION 9**

The Databricks workspace administrator has configured interactive clusters for each of the data engineering groups. To control costs, clusters are set to terminate after 30 minutes of inactivity. Each user should be able to execute workloads against their assigned clusters at any time of the day.

Assuming users have been added to a workspace but not granted any permissions, which of the following describes the minimal permissions a user would need to start and attach to an already configured cluster.

A. "Can Manage" privileges on the required cluster

B. Workspace Admin privileges, cluster creation allowed. "Can Attach To" privileges on the required cluster

C. Cluster creation allowed. "Can Attach To" privileges on the required cluster

D. "Can Restart" privileges on the required cluster

E. Cluster creation allowed. "Can Restart" privileges on the required cluster

Correct Answer: C

Explanation: This is the minimal permission a user would need to start and attach to an already configured cluster. Cluster creation allowed means that the user can create new clusters or start existing clusters that are stopped. "Can Attach To" privileges on the required cluster means that the user can attach notebooks or libraries to that cluster and run commands on it. Verified References: Databricks Certified Data Engineer Professional, under "Security and Governance" section; Databricks Documentation, under "Cluster permissions" section.

**QUESTION 10**

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".