CERT-PASS.com

Pass Your Cloud Cert. First Try.

Stop guessing. Start passing. Premium practice exams with real-world scenarios.

DATABRICKS | AWS | AZURE | GCP | SNOWFLAKE

- ✔ 93% First-Try Pass Rate
- ✔ Extensive Question Bank
- ✔ Updated for Latest Exams

THE ADVANTAGE

Exam-Accurate Content

Authored by certified engineers. Every question mirrors the real exam's structure, wording, and difficulty. No braindumps.

Deep Explanations

Don't just memorize. Every answer explains exactly "why A" is correct and "why not B, C, D" to build true comprehension.

Topic Analytics & Focus

Pinpoint your weak spots instantly with official blueprint-weighted scoring. Block distractions with Focus Mode.

Comprehensive Guides

Every single exam includes a fully structured curriculum. Learn the core concepts before diving into the practice tests.

THE PROCESS

01 **Pick Your Target Exam**

Browse our official catalog, select your certification, and generate your custom study plan.

02 **Practice & Analyze Data**

Take weighted practice exams. Get immediate feedback and watch readiness update in real time.

03 **Pass With Confidence**

Walk into the testing center backed by data, not hope. Join the 93% who pass on the first attempt.

Ready to get certified?

Start free today — no credit card required.

CERT-PASS

Databricks Data Engineer Associate

Free Practice Questions Preview

Here are 15 sample questions to help you get started. Unlock the full exam to access all 45+ questions with detailed explanations.

Question 1 : Data Processing & Transformations

A data organization leader is upset about the data analysis team's reports being different from the data engineering team's reports. The leader believes the siloed nature of their organization's data engineering and data analysis architectures is to blame. Which of the following describes how a data lakehouse could alleviate this issue?

A. Both teams would autoscale their work as data size evolves

B. Both teams would use the same source of truth for their work

C. Both teams would reorganize to report to the same department

D. Both teams would be able to collaborate on projects in real-time

E. Both teams would respond more quickly to ad-hoc requests

Answer: B

The correct answer is B: Both teams would use the same source of truth for their work.

A data lakehouse architecture unifies the best aspects of data lakes and data warehouses. Critically, it provides a single, consistent source of truth for both data engineers and data analysts. Currently, the leader

notes discrepancies between reports generated by the data analysis and data engineering teams, suggesting

these teams are operating on different data sets or transformations. A data lakehouse eliminates this discrepancy by providing a single repository for all data, accessible to both teams.

With a data lakehouse, data is ingested, transformed, and stored in a way that is consistent and accessible to

all stakeholders. Data engineers can focus on building pipelines to ingest and transform data, while data analysts can use the same data for reporting and analysis. This eliminates the need for data to be copied or

transformed multiple times, reducing the risk of errors and inconsistencies. Options A, C, D, and E are advantages of cloud architectures or organizational structures but don't directly address the core issue of differing reports due to separate data sources. The centralization of data and metadata within a unified platform is the key benefit a data lakehouse offers in resolving the leader's concern.

For further information, explore these resources:

1. Databricks Lakehouse Platform: https://www.databricks.com/product/data-lakehouse
2. What is a Data Lakehouse?: https://aws.amazon.com/what-is/data-lakehouse/

Question 2 : Data Processing & Transformations

Which of the following benefits of using the Databricks Lakehouse Platform is provided by Delta Lake?

A. An automated report needs to be refreshed as quickly as possible.

B. An automated report needs to be made reproducible.

C. An automated report needs to be tested to identify errors.

D. An automated report needs to be version-controlled across multiple collaborators.

E. An automated report needs to be runnable by all stakeholders.

Answer: A

The correct answer is D: The ability to support batch and streaming workloads. Delta Lake is a storage layer
that brings reliability to data lakes. A key characteristic of Delta Lake is its ACID (Atomicity, Consistency, Isolation, Durability) transaction support. This enables Delta Lake to handle both batch and streaming data reliably. Batch processing involves large volumes of data processed at once, whereas streaming processes
data in near real-time. Delta Lake's ability to handle both types of workloads provides flexibility. Options A, B,
and C are features of the Databricks platform but aren't directly enabled by Delta Lake. Option E describes a
general function that could be executed on the data itself, not a fundamental benefit provided by Delta Lake.
The ability to support both batch and streaming is a direct feature of the Delta Lake storage layer.
Further Research:
Databricks Delta Lake: https://www.databricks.com/product/delta-lake
Delta Lake Documentation: https://docs.databricks.com/delta/index.html

Question 3 : Data Processing & Transformations

Which of the following describes the storage organization of a Delta table?

A. Worker node

B. JDBC data source

C. Databricks web application

D. Databricks Filesystem

E. Driver node

Answer: C

The correct answer is C because Delta Lake, the technology behind Delta tables, is built on top of cloud storage like AWS S3, Azure Blob Storage, or Google Cloud Storage. It leverages the distributed nature of these object storage systems for scalability and reliability.

Delta Lake doesn't store all data in a single file. Instead, data is partitioned into multiple data files (typically Parquet format), and these files are stored in the underlying cloud storage. This allows for parallel processing
and efficient querying.
The "history" and "metadata" are critical components of Delta Lake's functionality, enabling features like time
travel, ACID transactions, and schema evolution. This metadata isn't stored within each data file but rather as
a transaction log alongside the data files. The transaction log is an ordered record of every operation performed on the Delta table.
Specifically, each commit to a Delta table creates a new entry (a metadata file) in the _delta_log directory, which is typically located alongside the data files in the same storage location. This log tracks changes to the
data, including additions, deletions, and updates. These metadata files (often stored in JSON format) contain
information about the data files involved in each transaction, enabling Delta Lake to track the table's state over time.
Therefore, a Delta table isn't a single file. It's a directory containing data files and a _delta_log directory containing metadata files. Answer C accurately reflects this distributed storage model, encompassing both the data and the necessary metadata for transaction management and data versioning. The distributed storage enables efficient processing and scaling provided by the underlying cloud storage layer. Options A, B,
D, and E are incorrect because they imply a single-file storage or an incomplete view of how metadata is managed within a Delta table.
Here are some authoritative links for further research:
1. Delta Lake Documentation (Databricks): https://docs.databricks.com/delta/index.html - This is the official documentation for Delta Lake and contains detailed information on its architecture and features.
2. Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores: https://databricks.com/blog/2019/04/24/delta-lake-high-performance-acid-table-storage-overcloud-object-stores.html - Databricks blog explaining the architecture and benefits of Delta Lake.
3. Understanding the Delta Lake Transaction Log: https://medium.com/datascale/understanding-thedelta-lake-transaction-log-416503e29f64 - A deep dive into how Delta Lake manages transactions
using the transaction log.

Question 4 : Data Processing & Transformations

Which of the following code blocks will remove the rows where the value in column age is greater than 25 from the existing Delta table my_table and save the updated table?

A. The ability to manipulate the same data using a variety of languages

B. The ability to collaborate in real time on a single notebook

C. The ability to set up alerts for query failures

D. The ability to support batch and streaming workloads

E. The ability to distribute complex data operations

Answer: D

The correct answer is C, DELETE FROM my_table WHERE age > 25;. This SQL command directly and efficiently removes rows from the my_table Delta table where the age column's value exceeds 25. Delta Lake,
built on Apache Spark, supports standard SQL DELETE operations for modifying data within Delta tables. This
functionality is crucial for data governance, regulatory compliance (like GDPR), and overall data quality management.
Option A (SELECT * FROM my_table WHERE age > 25;) only selects rows meeting the condition; it doesn't
modify the table. Options B and D use UPDATE, which is designed to modify existing values in specified columns. UPDATE isn't used for row deletion. Option E (DELETE FROM my_table WHERE age <= 25;) is incorrect because it deletes the wrong rows; it removes rows where the age is less than or equal to 25, which
is the opposite of what the question asks for.
The DELETE command with the WHERE clause provides the precise filtering logic required to remove the specific rows based on the given condition. Using Delta Lake transactions, this DELETE operation ensures atomicity, consistency, isolation, and durability (ACID) properties. If the DELETE command encounters an error, the entire operation is rolled back, maintaining data integrity. This contrasts sharply with non-ACID data
storage systems where partial updates might leave the data in an inconsistent state. Delta Lake also automatically maintains a version history, enabling auditing and rollback to previous versions if needed after
the DELETE operation.
For further reading on Delta Lake DELETE operations, refer to the official Databricks documentation:
Databricks Delta Lake Documentation: https://docs.databricks.com/delta/index.html
Delta Lake SQL Commands: https://docs.databricks.com/sql/language-manual/delta-lake-sql.html
These links provide comprehensive information on using Delta Lake and its SQL commands, including DELETE, with transaction guarantees.

Question 5 : Data Processing & Transformations

A data engineer has realized that they made a mistake when making a daily update to a table. They need to use Delta time travel to restore the table to a version that is 3 days old. However, when the data engineer attempts to time travel to the older version, they are unable to restore the data because the data files have been deleted. Which of the following explains why the data files are no longer present?

A. Delta tables are stored in a single file that contains data, history, metadata, and other attributes.

B. Delta tables store their data in a single file and all metadata in a collection of files in a separate location.

C. Delta tables are stored in a collection of files that contain data, history, metadata, and other attributes.

D. Delta tables are stored in a collection of files that contain only the data stored within the table.

E. Delta tables are stored in a single file that contains only the data stored within the table.

Answer: C

The reason the data engineer is unable to time travel to a version of the Delta table that is 3 days old is most
likely due to the VACUUM command having been run on the table. The VACUUM command removes data files
from the Delta Lake table's underlying storage that are no longer needed by the Delta Lake transaction log.
Its primary purpose is to reduce storage costs and improve query performance by eliminating outdated files.
By default, Delta Lake retains historical versions of the data for 30 days, allowing users to time travel to any
point within that window. However, if the VACUUM command is executed with a retention period shorter than
3 days (or if the default 7-day retention is already applied automatically), the data files required for the 3-dayold version would have been physically removed from the storage.
The TIME TRAVEL command itself is used to restore a table to a previous version, so it cannot be the cause.
The DELETE HISTORY command, while related to removing versions, is not a standard Delta Lake command.
Optimizing the table with the OPTIMIZE command reorganizes data files for better performance but doesn't
inherently delete historical data. Similarly, the HISTORY command simply displays the table's history and doesn't modify or remove any data files. Therefore, VACUUM, with its function of physically removing data files based on a retention policy, directly explains why the data files are no longer available for time travel to a
version older than the retention period. If VACUUM has been executed with a retention interval less than 3 days, restoring the table to a version 3 days old will be impossible, as the data files needed for that version are already purged.
Authoritative Links:
Databricks Delta Lake Vacuum: https://docs.databricks.com/delta/optimizations/vacuum.html
Databricks Delta Lake Time Travel: https://docs.databricks.com/delta/versioning.html

Question 6 : Data Processing & Transformations

Which of the following data lakehouse features results in improved data quality over a traditional data lake?

A. SELECT * FROM my_table WHERE age > 25;

B. UPDATE my_table WHERE age > 25;

C. DELETE FROM my_table WHERE age > 25;

D. UPDATE my_table WHERE age <= 25;

E. DELETE FROM my_table WHERE age <= 25;

Answer: C

The correct answer is B. A data lakehouse supports ACID-compliant transactions.

Here's why:

Data quality directly benefits from ACID (Atomicity, Consistency, Isolation, Durability) transactions. ACID properties ensure that data operations are reliable and consistent, preventing data corruption and maintaining
data integrity.

Atomicity: Guarantees that a transaction is treated as a single, indivisible unit of work. Either all changes within the transaction are applied, or none are. This prevents partial updates that could lead to inconsistent data.

Consistency: Ensures that a transaction moves the database from one valid state to another. Constraints, rules, and validations are enforced, preventing invalid data from being written.

Isolation: Dictates that concurrent transactions do not interfere with each other. Each transaction operates as
if it were the only transaction running on the system, preventing data corruption from concurrent updates.

Durability: Ensures that once a transaction is committed, it is permanently recorded and will survive system
failures (e.g., power outages, crashes).

Traditional data lakes often lack ACID transaction support, leading to "dirty reads" and inconsistent data during concurrent writes or failures. Data lakehouses, by incorporating ACID transactions, provide a more reliable and consistent foundation for data, thus improving data quality. Features like Delta Lake are used to
provide the ACID properties to Databricks Data Lakehouse.

The other options are less directly related to improved data quality compared to a traditional data lake, although they provide benefits to the general functionality. For example, storing data in open formats or using
SQL are also typical of a data lake, and they don't ensure the correctness of the data being stored.

Supporting Resources:

Databricks Documentation on Delta Lake (ACID Transactions): https://www.databricks.com/product/delta-lake

ACID Transactions in Data Warehousing: https://www.ibm.com/docs/en/db2/11.5?topic=concepts-acidproperties

Question 7 : Data Processing & Transformations

A data analyst has created a Delta table sales that is used by the entire data analysis team. They want help from the data engineering team to implement a series of tests to ensure the data is clean. However, the data engineering team uses Python for its tests rather than SQL. Which of the following commands could the data engineering team use to access sales in PySpark?

A. The VACUUM command was run on the table

B. The TIME TRAVEL command was run on the table

C. The DELETE HISTORY command was run on the table

D. The OPTIMIZE command was nun on the table

E. The HISTORY command was run on the table

Answer: A

E. spark.table("sales")

A Delta table created in Databricks (or any Spark environment using Delta Lake) is stored in the Metastore.

This means the table can be accessed both:

Through SQL:

SELECT * FROM sales;

And through PySpark using the SparkSession API.

To load a table in PySpark, the correct command is:

df = spark.table("sales")

This returns the Delta table as a DataFrame that the data engineering team can use for testing, transformations, or validation in Python.

Why the Other Options Are Incorrect:

A. SELECT * FROM sales

This is SQL only, not PySpark. It returns a result within the SQL interpreter, not a PySpark DataFrame.

B. "No way to share data"

Incorrect. Spark provides full interoperability between SQL and PySpark via the Metastore.

C. spark.sql("sales")

Incorrect. spark.sql() expects a full SQL query. The correct usage would require: spark.sql("SELECT * FROM

sales").

D. spark.delta.table("sales")

Not valid syntax. The correct form would be: deltaTable = DeltaTable.forName(spark, "sales"), which returns a

DeltaTable object (not a DataFrame).

Question 8 : Data Processing & Transformations

A data engineer wants to create a new table containing the names of customers that live in France. They have written the following command: A senior data engineer mentions that it is organization policy to include a table property indicating that the new table includes personally identifiable information (PII). Which of the following lines of code fills in the above blank to successfully complete the task?

A. Commit

B. Pull

C. Push

D. Clone

E. Merge

Answer: E

D. COMMENT "Contains PII"

In SQL (especially when working with data platforms like Databricks, Hive, or Spark SQL), metadata can be

added to tables using the COMMENT clause to describe the contents or purpose of the table.

If your organization requires tagging tables that contain Personally Identifiable Information (PII), adding a comment like "Contains PII" is a standard and accepted way to do it.

Question 9 : Data Processing & Transformations

Which of the following benefits is provided by the array functions from Spark SQL?

A. A data lakehouse provides storage solutions for structured and unstructured data.

B. A data lakehouse supports ACID-compliant transactions.

C. A data lakehouse allows the use of SQL queries to examine data.

D. A data lakehouse stores data in open formats.

E. A data lakehouse enables machine learning and artificial Intelligence workloads.

Answer: B

The correct answer is D because Spark SQL's array functions are specifically designed to handle complex, nested data structures, commonly encountered when ingesting data from sources like JSON files. JSON often

contains arrays of values or nested objects, which can be challenging to process with traditional SQL. Array

functions provide the means to manipulate and extract information from these array structures, such as accessing elements by index, transforming array contents, and performing operations on array elements.

Options A, B, and C are less relevant to the primary purpose of array functions. While Spark SQL can handle a

variety of data types (A), that's not the exclusive domain of array functions. Window functions (B) handle calculations across sets of rows, not array manipulation, and date/time functions (C) are designed for timeseries data, not array processing. Option E refers to handling an array of tables, which can be accomplished

via other Spark SQL features like UNION or dynamic views, but doesn't directly represent the capability of array functions.

Therefore, the most direct and accurate benefit offered by Spark SQL's array functions is the ability to efficiently process and transform complex, nested data structures, especially those originating from JSON or

similar semi-structured formats. They simplify the task of working with these intricate data structures and provide a more accessible way to extract insights from the data.

Reference:

Databricks documentation on array functions:

https://docs.databricks.com/sql/languagemanual/functions/array_contains.html (Example - illustrates

usage)
Apache Spark SQL documentation: https://spark.apache.org/docs/latest/sql-ref.html (Provides a complete overview of all SQL functions, including array functions)

Question 10 : Data Processing & Transformations

Which of the following commands can be used to write data into a Delta table while avoiding the writing of duplicate records?

A. Databricks Repos automatically saves development progress

B. Databricks Repos supports the use of multiple branches

C. Databricks Repos allows users to revert to previous versions of a notebook

D. Databricks Repos provides the ability to comment on specific changes

E. Databricks Repos is wholly housed within the Databricks Lakehouse Platform

Answer: B
The correct answer is C. MERGE. Here's a detailed justification:
The MERGE command in Delta Lake is specifically designed for performing upserts - combining update and
insert operations based on a specified condition. This is precisely what's needed to avoid writing duplicate records. The MERGE command essentially allows you to specify a source dataset (the new data you're trying
to write) and a target Delta table. You then define a condition that determines when a row from the source matches a row in the target.
If a match is found based on this condition, you can specify how to update the existing row in the target table
(the "update" part of upsert). If no match is found, you can specify how to insert the new row from the source
into the target table (the "insert" part of upsert).
By using MERGE with an appropriate matching condition (e.g., based on a unique key or combination of fields),
you can effectively prevent the creation of duplicate records. If a record with the same key already exists, it will be updated; otherwise, a new record will be inserted. This ensures that your Delta table maintains data integrity.
Options A, B, D, and E are incorrect:
DROP: This command is used to delete entire tables or partitions, not to avoid writing duplicates during data
ingestion.
IGNORE: While some database systems might have an IGNORE option, it's not a standard Delta Lake or Spark
command for handling duplicate records. It wouldn't provide the necessary logic for an upsert operation.
APPEND: This command simply adds new data to the end of the table without checking for duplicates. It's the

opposite of what we need.
INSERT: Similar to APPEND, INSERT adds new rows without any de-duplication logic. Using INSERT alone is a
guaranteed way to potentially introduce duplicate records.
In summary, MERGE provides the fine-grained control needed to update existing records or insert new ones
only when necessary, avoiding duplication and maintaining data integrity within your Delta table.
Further reading on Delta Lake MERGE can be found here:
Databricks Documentation: MERGE INTO

Question 11 : Data Processing & Transformations

A data engineer needs to apply custom logic to string column city in table stores for a specific use case. In order to apply this custom logic at scale, the data engineer wants to create a SQL user-defined function (UDF). Which of the following code blocks creates this SQL UDF?

A. Databricks account representative

B. This transfer is not possible

C. Workspace administrator

D. New lead data engineer

E. Original data engineer

Answer: C
A is the correct answer. A SQL UDF in Databricks uses CREATE FUNCTION with RETURNS (type) followed by RETURN (expression). Option D is missing the RETURNS clause. Option E uses CREATE UDF which is not valid SQL syntax — it must be CREATE FUNCTION. Options B and C are Python UDFs, not SQL UDFs.

Question 12 : Data Processing & Transformations

A data engineering team has two tables. The first table march_transactions is a collection of all retail transactions in the month of March. The second table april_transactions is a collection of all retail transactions in the month of April. There are no duplicate records between the tables. Which of the following commands should be run to create a new table all_transactions that contains all records from march_transactions and april_transactions without duplicate records?

A. SELECT * FROM sales

B. There is no way to share data between PySpark and SQL.

C. spark.sql("sales")

D. spark.delta.table("sales")

E. spark.table("sales")

Answer: E

The correct command to combine all records from march_transactions and april_transactions into a new table
all_transactions without duplicates is option B: CREATE TABLE all_transactions AS SELECT FROM march_transactions UNION SELECT FROM april_transactions;
Here's why:
UNION: The UNION operator in SQL is specifically designed to combine the result-sets of two or more SELECT statements. Importantly, UNION automatically removes duplicate rows from the final result, ensuring
each record is unique. This directly addresses the requirement of creating a table without duplicate records
from the two source tables.
INNER JOIN, OUTER JOIN, INTERSECT, MERGE: These other options are not suitable for combining records
from two separate tables in the way described in the prompt.
INNER JOIN and OUTER JOIN are used to combine rows from two tables based on a related column. Since the
goal is to combine all rows regardless of a shared key, these are incorrect.
INTERSECT returns only the rows that are present in both tables, not all rows from both.
MERGE is typically used for complex data manipulations, such as synchronizing two tables based on a condition, and it is not a standard SQL command in all database systems. Databricks may support it through
variations in spark SQL, but it is not the simplest solution.
CREATE TABLE AS SELECT (CTAS): The CREATE TABLE all_transactions AS SELECT ... syntax allows
creating a new table based on the results of a SELECT query. This is an efficient way to directly populate the
new table with the combined and de-duplicated data.
Therefore, the UNION operator within a CREATE TABLE AS SELECT statement is the most appropriate way to
create the all_transactions table with unique records from both input tables.
Supporting Documentation:
Databricks SQL Reference - UNION: https://docs.databricks.com/sql/language-manual/sql-ref-syntax-qryselect-union.html
Databricks SQL Reference - CREATE TABLE AS SELECT:
https://docs.databricks.com/sql/languagemanual/sql-ref-syntax-ddl-create-table-as-select.html

Question 13 : Data Processing & Transformations

A data engineer only wants to execute the final block of a Python program if the Python variable day_of_week is equal to 1 and the Python variable review_period is True. Which of the following control flow statements should the data engineer use to begin this conditionally executed code block?

A. DESCRIBE LOCATION customer360;

B. DROP DATABASE customer360;

C. DESCRIBE DATABASE customer360;

D. ALTER DATABASE customer360 SET DBPROPERTIES ('location' = '/user' ;

E. USE DATABASE customer360;

Answer: C

The correct answer is D: if day_of_week == 1 and review_period:.

The if statement in Python is used to execute a block of code only if a certain condition is met. This condition is

evaluated as a boolean (True or False). In this scenario, the data engineer wants the code to execute only when day_of_week is equal to 1 and review_period is True.

Let's break down why other options are incorrect:

A. if day_of_week = 1 and review_period:: The = operator is an assignment operator, not a comparison operator. Using it in an if statement like this would attempt to assign the value 1 to day_of_week, which is syntactically incorrect and will raise an error. You need the comparison operator == to check for equality.

B. if day_of_week = 1 and review_period = "True":: This suffers from the same assignment error as option A.

Additionally, even if corrected to use ==, comparing a boolean variable review_period to the string "True" is

generally incorrect and will likely not produce the desired result. A boolean variable is already True or False.

C. if day_of_week == 1 and review_period == "True":: Correctly uses the == operator, however this incorrectly compares a boolean variable review_period to a string "True". This will only evaluate True if review_period is the string value "True" which is unlikely to be the case.

E. if day_of_week = 1 & review_period: = "True":: Contains both the assignment error (=) and uses the bitwise

AND operator (&) instead of the logical AND operator (and). Additionally, has the same boolean comparison

issue as option B and C.

Option D, if day_of_week == 1 and review_period:, correctly uses the equality operator == to compare day_of_week to 1. It also correctly checks if review_period is True. Because review_period is already a boolean

variable (either True or False), simply stating its name after the and implicitly checks if it's True. The whole condition will only be true if both conditions are satisfied.

In short, using and ensures that both conditions, day_of_week == 1 and review_period, must be True for the

conditional code block to execute.

Relevant links for more information:

Python if statement: https://www.w3schools.com/python/python_conditions.asp

Python comparison operators: https://www.w3schools.com/python/python_operators.asp

Python boolean data type: https://www.w3schools.com/python/python_booleans.asp

Question 14 : Data Processing & Transformations

A data engineer is attempting to drop a Spark SQL table my_table. The data engineer wants to delete all table metadata and data. They run the following command: DROP TABLE IF EXISTS my_table While the object no longer appears when they run SHOW TABLES, the data files still exist. Which of the following describes why the data files still exist and the metadata files were deleted?

A. There is no way to indicate whether a table contains PII.

B. "COMMENT PII"

C. TBLPROPERTIES PII

D. COMMENT "Contains PII"

E. PII

Answer: D

The reason the data files still exist despite dropping the table is that the table was an external table. In Spark
SQL, when you drop a managed table (also known as an internal table), Spark removes both the metadata and
the underlying data from the metastore and storage location, respectively. However, when you drop an external table, only the metadata is removed from the metastore. The data files themselves, residing in the external storage location (e.g., cloud object storage like AWS S3 or Azure Blob Storage), remain untouched.
The DROP TABLE command only removes the table's definition from the metastore, disconnecting the table
name from the actual data location. The IF EXISTS clause simply prevents an error if the table does not exist,
and it doesn't affect the behavior regarding data deletion. The size of the data (options A and B) is irrelevant.
Option D, "The table did not have a location," is contradictory, as external tables are defined by their location,
unlike managed tables that default to the metastore's managed location. In summary, the defining characteristic of external tables is that their data lives outside the control of the metastore and is unaffected
by table dropping operations.
Further research can be done here:https://spark.apache.org/docs/latest/sql-data-sources-hivetables.htmlhttps://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-ddl-drop-table.html

Question 15 : Data Processing & Transformations

A data engineer wants to create a data entity from a couple of tables. The data entity must be used by other data engineers in other sessions. It also must be saved to a physical location. Which of the following data entities should the data engineer create?

A. An ability to work with data in a variety of types at once

B. An ability to work with data within certain partitions and windows

C. An ability to work with time-related data in specified intervals

D. An ability to work with complex, nested data ingested from JSON files

E. An ability to work with an array of tables for procedural automation

Answer: D

The correct answer is E. Table. Here's why:

The requirement is to create a data entity accessible across multiple sessions and persistently stored in a physical location. Let's examine each option:

A. Database: A database is a container for schemas, tables, views, and functions. While necessary for organizing data, it doesn't directly represent a data entity derived from other tables. It's more like the overall
storage structure.

B. Function: Functions are reusable code blocks, not data entities derived from tables. They perform operations, not store data.

C. View: Views are virtual tables based on the result-set of a query. While they can combine data from multiple tables, they are not physically stored. Views are simply stored queries.

D. Temporary View: Temporary views are session-scoped. This means they exist only for the duration of the
Databricks session in which they were created. This directly contradicts the requirement of being accessible
across multiple sessions.

E. Table: Tables are physical representations of data stored in a structured format. They persist across sessions and are stored in a defined physical location (e.g., cloud storage like AWS S3 or Azure Data Lake
Storage). A table can be created by selecting data from other tables, fulfilling the data engineer's requirement to combine data. Thus, other data engineers can access it in other sessions. This makes it the
correct option. A managed table stores the data and metadata in the metastore while an unmanaged table only stores metadata in the metastore, and references the physical storage location that is manually defined.

In summary, only a table provides the required combination of persistence, cross-session availability, and physical storage for the derived data entity.

Further Research:

Databricks Tables: https://docs.databricks.com/en/tables/index.html

Databricks Views: https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-aux-show-views.html