



**Databricks Certified Associate Developer for Apache Spark 3.5-Python**

**Databricks Databricks-Certified-Associate-Developer-for-Apache-Spark-3.5**

**Version Demo**

**Total Demo Questions: 2**

**Total Premium Questions: 23**

**Buy Premium PDF**

**<https://passqueen.com>**

**[support@passqueen.com](mailto:support@passqueen.com)**

**PASSQUEEN.COM**

**[support@passqueen.com](mailto:support@passqueen.com)**

## Topic Break Down

| Topic                              | No. of Questions |
|------------------------------------|------------------|
| Topic 1, Apache Spark Architecture | 3                |
| Topic 2, Apache Spark DataFrames   | 14               |
| Topic 3, Data Processing           | 3                |
| Topic 4, Data Writing              | 3                |
| Total                              | 23               |

#### QUESTION NO: 1

Which command overwrites an existing JSON file when writing a DataFrame?

- A. `df.write.mode("overwrite").json("path/to/file")`
- B. `df.write.overwrite.json("path/to/file")`
- C. `df.write.json("path/to/file", overwrite=True)`
- D. `df.write.format("json").save("path/to/file", mode="overwrite")`

**ANSWER: A**

#### Explanation:

`df.write.mode("overwrite").json("path/to/file")` is the standard PySpark `DataFrameWriter` command for writing a DataFrame as JSON while replacing data already present at the target path. The `write` property returns a `DataFrameWriter`, and `mode("overwrite")` sets the save mode before the output operation is invoked. The subsequent `json("path/to/file")` call selects JSON as the output data source and writes to the supplied location. In `overwrite` mode, Spark replaces the existing output at that path rather than raising an error because the destination already exists. This syntax is concise and directly communicates both required choices: the write behavior and the JSON output format. Spark writes JSON output as a directory containing one or more part files, rather than necessarily as one standalone file, so the supplied path should be understood as the output location. The PySpark API documents `DataFrameWriter.mode` and its supported `overwrite` mode at [the PySpark DataFrameWriter.mode reference](#). JSON-specific writing behavior is documented at [the PySpark DataFrameWriter.json reference](#).

#### QUESTION NO: 2

A DataFrame contains a column named `tags` with type `array<string>`. The engineer needs one output row for every tag while retaining the other columns from the original row. Rows where `tags` is an empty array should not produce an output row.

Which transformation should the engineer use?

- A. `df.withColumn("tag", explode("tags"))`
- B. `df.withColumn("tag", array_distinct("tags"))`
- C. `df.withColumn("tag", split("tags", ","))`
- D. `df.select("*", posexplode("tags"))`

**ANSWER: A**

#### Explanation:

`explode("tags")` creates a separate row for each array element and places the element in the specified output column. An empty array produces no rows, which matches the requirement.

`array_distinct()` removes repeated values but leaves the data as an array in one row. `split()` converts a string into an array and is not appropriate for an existing array column. `posexplode()` also expands the array but returns both an element position and an element, adding an unnecessary column.