databricks

Databricks Exam Guide

Databricks Certified Data Engineer Professional

Provide Exam Guide Feedback

Purpose of this Exam Guide

This exam guide gives you an overview of the Databricks Certified Data Engineer Professional exam and what it covers to help you determine your exam readiness. This document will get updated anytime there are any changes to an exam (and when those changes will take effect on an exam), so that you can be prepared. **Please check back two weeks before your exam to ensure you have the most current version. This version covers the currently live version as of July 3, 2026.**

Audience Description

The Databricks Certified Data Engineering Professional exam validates a candidate's advanced skills in building, optimizing, and maintaining production-grade data engineering solutions on the Databricks Data Intelligence Platform. Successful candidates demonstrate expertise across core platform features such as Delta Lake, Unity Catalog, Auto Loader, Lakeflow Spark Declarative Pipelines, Databricks Compute (including serverless), Lakeflow Jobs and the Medallion Architecture. This Certification assesses the ability to design secure, reliable, and cost-effective ETL Pipelines, process complex data from diverse sources using Python and SQL, and apply best practices in schema management, observability, governance, and performance optimization. Candidates are also tested on implementing streaming workloads, orchestrating workflows, leveraging DevOps & CI/CD, and deploying with tools like the Databricks CLI, REST API, and Asset Bundles. Individuals who pass this certification exam can be expected to complete advanced data engineering tasks using Databricks and its associated tools.

About the Exam

- Number of items: 59 scored multiple-choice questions
- Time limit: 120 minutes
- Registration fee: USD 200, plus applicable taxes as required per local law
- Delivery method: Online Proctored and test center proctored
- Test Aides: No Test Aides provided, including API Documentation.
- Prerequisite: None is required; related course attendance and one year of hands-on experience in Data Engineering tasks outlined in the exam guide are highly recommended.
- Validity: 2 years
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the current version of the exam. Please review the "Getting Ready for the Exam" section below to prepare for your recertification exam.

- Unscored Content: Exams may include unscored items to gather statistical information for future use. These items are not identified on the form and do not impact your score, and additional time is factored into account for this content.

Recommended Training

- Instructor-led: Advanced Data Engineering With Databricks
- Self-paced (available in Databricks Academy):
    - Advanced Techniques with Spark Declarative Pipeline
    - Databricks Data Privacy
    - Databricks Performance Optimization
    - Automated Deployment with Declarative Automation Bundles

Exam outline

Section 1: Developing Code for Data Processing using Python and SQL

- Using Python and Tools for development
    - Design and implement a scalable Python project structure optimized for Declarative Automation Bundles (formerly Databricks Asset Bundles / DABs), enabling modular development, deployment automation, and CI/CD integration.
    - Manage and troubleshoot external third-party library installations and dependencies in Databricks, including PyPI packages, local wheels, and source archives.
    - Develop User-Defined Functions (UDFs) using Pandas/Python UDF.
- Building and Testing an ETL pipeline with Lakeflow Spark Declarative Pipelines, SQL, and Apache Spark on the Databricks Platform
    - Build and manage reliable, production-ready data pipelines for batch and streaming data using Lakeflow Spark Declarative Pipelines and Autoloader.
    - Create and Automate ETL workloads using Jobs via UI/APIs/CLI.
    - Explain the advantages and disadvantages of streaming tables compared to materialized views.
    - Use AUTO CDC APIs (formerly APPLY CHANGES) to simplify CDC in Lakeflow Spark Declarative Pipelines.
    - Compare Spark Structured Streaming and Lakeflow Spark Declarative Pipelines to determine the optimal approach for building scalable ETL pipelines.
    - Create a pipeline component that uses control flow operators (e.g., if/else, for/each, etc.).
    - Choose the appropriate configs for environments and dependencies, high memory for notebook tasks, and auto-optimization to disallow retries.
    - Develop unit and integration tests using assertDataFrameEqual, assertSchemaEqual, DataFrame.transform, and testing frameworks, to ensure code correctness, including a built-in debugger.

Section 2: Data Ingestion & Acquisition:

- Design and implement data ingestion pipelines to efficiently ingest a variety of data formats including Delta Lake, Parquet, ORC, AVRO, JSON, CSV, XML, Text and Binary from diverse sources such as message buses and cloud storage.
- Create an append-only data pipeline capable of handling both batch and streaming data using Delta.

Section 3: Data Transformation, Cleansing, and Quality

- Write efficient Spark SQL and PySpark code to apply advanced data transformations, including window functions, joins, and aggregations, to manipulate and analyze large Datasets.
- Develop a quarantining process for bad data with Lakeflow Spark Declarative Pipelines, or autoloader in classic jobs.

Section 4: Data Sharing and Federation

- Demonstrate delta sharing securely between Databricks deployments using Databricks to Databricks Sharing (D2D) or to external platforms using the open sharing protocol (D2O).
- Configure Lakehouse Federation with proper governance across the supported source Systems.
- Use Delta Share to share live data from Lakehouse to any computing platform.

Section 5: Monitoring and Alerting

- Monitoring
    - Use system tables for observability over resource utilization, cost, auditing and workload monitoring.
    - Use Query Profiler UI and Spark UI to monitor workloads.
    - Use the Databricks REST APIs/Databricks CLI for monitoring jobs and pipelines.
    - Use Lakeflow Spark Declarative Pipelines Event Logs to monitor pipelines.
- Alerting
    - Use SQL Alerts to monitor data quality.
    - Use the Lakeflow Jobs UI and Jobs API to set up notifications for job status and performance issues.

Section 6: Cost & Performance Optimization

- Understand how / why using Unity Catalog managed tables reduces operations
- Overhead and maintenance burden.
- Understand delta optimization techniques, such as deletion vectors and liquid clustering.
- Understand the optimization techniques used by Databricks to ensure the performance of queries on large datasets (data skipping, file pruning, etc.).

- Apply Change Data Feed (CDF) to address specific limitations of streaming tables and enhance latency.
- Use the query profile to analyze the query and identify bottlenecks, such as bad data skipping, inefficient types of joins, and data shuffling.

Section 7: Ensuring Data Security and Compliance

- Applying Data Security mechanisms.
    - Use ACLs to secure Workspace Objects, enforcing the principle of least privilege, including enforcing principles like least privilege, policy enforcement.
    - Use row filters and column masks to filter and mask sensitive table data.
    - Apply anonymization and pseudonymization methods, such as Hashing, Tokenization, Suppression, and generalization, to confidential data.
- Ensuring Compliance
    - Implement a compliant batch & streaming pipeline that detects and applies masking of PII to ensure data privacy.
    - Develop a data purging solution ensuring compliance with data retention policies.

Section 8: Data Governance

- Create and add descriptions/metadata about enterprise data to make it more discoverable.
- Demonstrate understanding of Unity Catalog permission inheritance model.

Section 9: Debugging and Deploying

- Debugging and Troubleshooting
    - Identify pertinent diagnostic information using Spark UI, cluster logs, system tables, and query profiles to troubleshoot errors.
    - Analyze the errors and remediate the failed job runs with job repairs and parameter overrides.
    - Use Lakeflow Spark Declarative Pipelines event logs and the Spark UI to debug Lakeflow Spark Declarative Pipelines and Spark pipelines.
- Deploying CI/CD
    - Build and deploy Databricks resources using Declarative Automation Bundles (formerly Databricks Asset Bundles)
    - Configure and integrate with Git-based CI/CD workflows Databricks Git folders (formerly Repos) for notebook and code deployment.

Section 10: Data Modeling

- Design and implement scalable data models using Delta Lake to manage large datasets.
- Simplify data layout decisions and optimize query performance using Liquid Clustering.
- Identify the benefits of using liquid Clustering over Partitioning and ZOrder.

- Design Dimensional Models for analytical workloads, ensuring efficient querying and aggregation.

Sample Questions

These questions are retired from a previous version of the exam. The purpose is to show you the objectives as they are stated on the exam guide, and give you a sample question that aligns with the objective. The exam guide lists the objectives that could be covered on an exam. The best way to prepare for a certification exam is to review the exam outline in the exam guide.

Question 1
Objective: Understand Delta Lake's catalog- metastore operations and ACID compliance behavior.

A Delta Lake table was created with the following query:

```
CREATE TABLE prod.sales_by_stor
USING DELTA
LOCATION "/mnt/prod/sales_by_store"
```

Realizing that the original query had a typographical error, the code below was executed:

```
ALTER TABLE prod.sales_by_stor RENAME TO prod.sales_by_store
```

Which result will occur after running the second command?

A. All related files and metadata are dropped and recreated in a single ACID transaction.
B. The table name change is recorded in the Delta transaction log.
C. A new Delta transaction log is created for the renamed table..
D. The table reference in the metastore is updated.

Question 2
Objective: Understand Spark Structured Streaming behaviour and determine the optimal approach for production SLA-ready pipelines.

A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. Currently, during normal execution, each microbatch of data is processed in under 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a 10-second trigger interval.

Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

A. Use the trigger once option and configure a Databricks job to execute the query every 8 seconds; this ensures all backlogged records are processed with each batch.
B. Decrease the trigger interval to 5 seconds; triggering batches more frequently may prevent records from backing up and large batches from causing a spill.
C. Decrease the trigger interval to 5 seconds; triggering batches more frequently allows

idle executors to begin processing the next batch while longer-running tasks from previous batches finish.

D. The trigger interval cannot be modified without modifying the checkpoint directory; to maintain the current stream state, increase the number of shuffle partitions to maximize parallelism.

Question 3
Objective: Design and implement scalable data models using Delta Lake to manage large datasets.

A Delta Lake table representing metadata about content posts from users has the following `schema:`

```
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT,
latitude FLOAT, post_time TIMESTAMP, date DATE
```

Based on the above schema, which column should be used for partitioning the Delta Table?

A. `post_id`
B. `post_time`
C. `date`
D. `user_id`

Question 4
Objective: Demonstrate understanding of Unity Catalog permission inheritance model

A table named `user_ltv` is being used to create a view that will be used by data analysts on various teams. Users in the workspace are configured into groups, which are used for setting up data access using ACLs.

The `user_ltv` table has the following schema:

```
email STRING, age INT, ltv INT
```

The following view definition is executed:

```
CREATE VIEW email_ltv AS
SELECT
CASE WHEN
  is_member('marketing') THEN email
  ELSE 'REDACTED'
END AS email,
ltv
FROM user_ltv
```

An analyst who is not a member of the marketing group executes the following query:

```
SELECT * FROM email_ltv
```

What will be the result of this query?

A. Only the **`email`** and **`ltv`** columns will be returned; the `email` column will contain the string "REDACTED" in each row.
B. Three columns will be returned, but one column will be named "REDACTED" and contain only null values.
C. Only the **`email`** and **`ltv`** columns will be returned; the **`email`** column will contain all null values.
D. The **`email`** and **`ltv`** columns will be returned with the values in **`user_ltv`**.

Question 5
Objective- Choose the appropriate configs for environments and dependencies, high memory for notebook tasks and auto-optimization to disallow retries.

The business reporting team requires that data for their dashboards be updated every hour. The total processing time for the pipeline, which extracts, transforms, and loads data for its runs, is 10 minutes.

Assuming normal operating conditions, which configuration will meet their service-level agreement requirements with the lowest cost?

A. Schedule a job to execute the pipeline once an hour on a dedicated interactive cluster.
B. Schedule a job to execute the pipeline once an hour on a new job cluster.
C. Schedule a Structured Streaming job with a trigger interval of 60 minutes.
D. Configure a job that executes every time new data lands in a given directory.

Question 6
Objective- Understand the Notebook development environment, variable management and creating secure, configurable code.

The security team is exploring whether the Databricks secrets module can be leveraged for connecting to an external database.

After testing the code with all Python variables being defined with strings, they upload the password to the secrets module and configure the correct permissions for the currently active user. They modify their code to the following (leaving all other variables unchanged).

```
password = dbutils.secrets.get(scope="db_creds", key="jdbc_password")

print(password)

df = (spark
  .read
  .format("jdbc")
  .option("url", connection)
  .option("dbtable", tablename)
  .option("user", username)
  .option("password", password)
  )
```

What will happen when this code is executed?

A. The connection to the external table will succeed; the string "REDACTED" will be printed.
B. The connection to the external table will succeed; the string value of the password will be printed in plain text.
C.. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed, and the password will be printed in plain text.
D. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed, and the encoded password will be saved to DBFS.

Question 7
Objective: Understand the optimization techniques used by Databricks to ensure performance of queries on large datasets (data skipping, file pruning, etc)

A data ingestion task requires a 1-TB JSON dataset to be written out to Parquet with a target part-file size of 512 MB. Because Parquet is being used instead of Delta Lake, built-in file-sizing features such as Auto-Optimize & Auto-Compaction cannot be used.

Which approach will work without rearranging the data?

A. Ingest the data, execute the narrow transformations, repartition to 2,048 partitions (1TB*1024*1024/512), and then write to Parquet.
B. Set `spark.sql.adaptive.advisoryPartitionSizeInBytes` to 512 MB, ingest the data, execute the narrow transformations, coalesce to 2,048 partitions (1TB*1024*1024/512), and then write to Parquet.
C. Set `spark.sql.files.maxPartitionBytes` to 512 MB, ingest the data, execute the narrow transformations, and then write to Parquet.
D. Set `spark.sql.shuffle.partitions` to 2,048 partitions (1TB*1024*1024/512), ingest the data, execute the narrow transformations, optimize the data by sorting it (which automatically repartitions the data), and then write to Parquet.

Question 8
Objective: Apply Delta Lake clone to learn how shallow and deep clones interact with source/target tables.

The marketing team wants to share data in an aggregate table with the sales organization, but the field names used by the teams do not match, and a number of marketing-specific fields have not been approved for the sales organization.

Which solution addresses the situation while emphasizing simplicity?

A. Create a view on the marketing table selecting only those fields approved for the sales team; alias the names of any fields that should be standardized to the sales naming

conventions.

B. Create a new table with the required schema and use Delta Lake's DEEP CLONE functionality to sync up changes committed to one table to the corresponding table.
C. Use a CTAS statement to create a derivative table from the marketing table, and then configure a production job to propagate the changes.
D. Add a parallel table write to the current production pipeline, updating a new sales table that varies as required from the marketing table.

Question 9
Objective: Create a multi-task job with multiple dependencies.

A Databricks job has been configured with three tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on task A.

What will be the resulting state if tasks A and B complete successfully but task C fails during a scheduled run?

A. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; some operations in task C may have been completed successfully.
B. Unless all tasks are completed successfully, no changes will be committed to the Lakehouse; because task C failed, all commits will be rolled back automatically.
C. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; any changes made in task C will be rolled back due to task failure.
D. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.

Answers

Question 1: D
Question 2: B
Question 3: C
Question 4: A
Question 5: B
Question 6: A
Question 7: C
Question 8: A
Question 9 :A