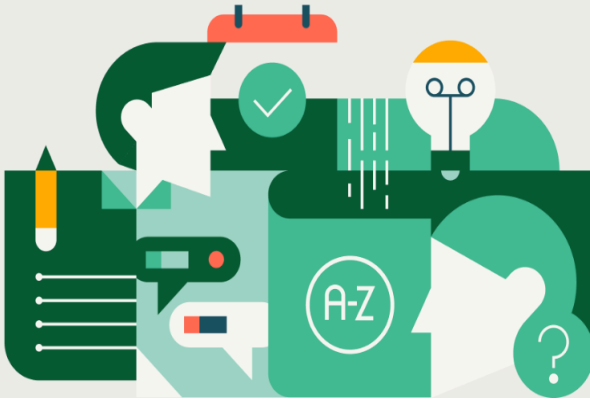
databricks

Databricks Exam Guide

# Databricks Certified Data Engineer Associate Renewal Exam




Provide Exam Guide Feedback

## Purpose of this Exam Guide

The purpose of this exam guide is to give you an overview of the exam and what is covered on the exam to help you determine your exam readiness. This document will be updated whenever there are changes to an exam (and when those changes take effect), so you can be prepared. This covers the exam version as of **Jun 15, 2026**.

## Audience Description

This Databricks Certified Data Engineer Associate Renewal exam assesses an individual's ability to utilize the Databricks Data Intelligence Platform to execute foundational data engineering tasks. The exam assesses a Databricks Certified Data Engineer Associate's knowledge of the tasks related to Data Ingestion, Data Loading, Data Transformation, and Modeling- such as the ability to perform Extract, Transform, Load (ETL) tasks using PySpark / SQL, working with Lakeflow Jobs, and CI/CD. Finally, the exam assesses the test takers' understanding of troubleshooting, monitoring, and optimization techniques, as well as their knowledge of achieving Governance and Security within the Databricks Platform.

## About the Exam

- Number of scored items: 25 scored multiple-choice questions.
- Time limit: 60 minutes.
- Registration fee: 200 USD
- Delivery method: Online Proctored Only
- Test aides: none allowed.
- Prerequisites: This renewal exam is available to individuals who hold an existing Databricks Certified Data Engineer Associate credential that has either expired or is within 2 months of expiration. It is highly recommended that candidates complete the suggested training and have at least 12 months of hands-on experience with the Databricks Platform.
- Validity: 2 years.
- Recertification:  Candidates must renew their credentials every 2 years by successfully completing the active exam or renewal exam. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam.

## Recommended Training

- Instructor-led: Data Engineering with Databricks
- Self-paced (available in Databricks Academy):
    - Data Ingestion with Lakeflow Connect
    - Deploy Workloads with Lakeflow Jobs
    - DevOps Essentials for Data Engineering
    - Data Interoperability with Unity Catalog
    - Build Data Pipelines with Lakeflow Spark Declarative pipeline
    - Get Started with Data Governance on Databricks

## Exam outline

**Section 1: Data Ingestion and Loading (32%)**

- Use COPY INTO to perform incremental file ingestion from cloud object storage into Unity Catalog-governed Delta tables.
- Use Auto Loader to incrementally ingest files into Unity Catalog-governed Delta tables
- Configure Lakeflow Connect to reliably ingest data from diverse enterprise sources into Unity Catalog–governed tables.
- Use JDBC/ODBC or REST clients in notebooks to land data into cloud storage or directly into Unity Catalog–governed tables.
- Prioritize between Auto Loader, Lakeflow Connect partner connectors, and other ingestion methods based on volume, frequency, data type, and governance needs.
- Ingest semi-structured and unstructured data (e.g. JSON and nested data) via Lakeflow Connect and managed connectors into Delta tables.

**Section 2: Data Transformation and Modeling (24%)**

- Apply Data Cleaning, reading bronze tables with PySpark/SQL (handle nulls, standardize data types), and write the results to silver tables.
- Identify common Spark tuning parameters and understand their general effect on performance.
- Differentiate Gold-layer objects in Unity Catalog (materialized views, views, streaming tables, regular tables) and their use for BI/analytics.
- Apply Data quality checks and validation rules to ensure reliable Silver and Gold datasets

**Section 3: Working with Lakeflow Jobs (12%)**

- Implement control flows (task retries, branching, looping) using Lakeflow Jobs to orchestrate multi-step data workflows
- Configure common task types and dependencies in Lakeflow Jobs using a DAG-based workflow
- Configure Lakeflow Jobs schedules and triggers, including scheduled, file arrival, table

update, and continuous triggers.

**Section 4: Implementing CI/CD (12%)**

- Manage code in Databricks workspace UI using Git Folders: create/switch branches, commit, push, and open pull requests.
- Use Declarative Automation Bundle environment targets (dev, test, prod) to promote a codebase across environments.
- Use the Databricks CLI to validate and deploy Declarative Automation Bundles (DABs)

**Section 5: Troubleshooting, Monitoring, and Optimization (12%)**

- Identify trends in job performance using the Lakeflow Jobs run history view.
- Use the Lakeflow Jobs UI to monitor pipeline health, job statuses, DAG task graphs, run times, and failure rates.
- Identify the purpose and benefits of Liquid Clustering and predictive optimization as managed Delta Lake features.

**Section 6: Governance and Security (8%)**

- Understand the purpose of column-level masking and row-level security and identify when they should be applied.
- Understand the features of Unity Catalog ABAC policies and how they support centralized data access control.

## Sample Questions

These questions are retired from a previous version of the exam. The purpose is to show you the objectives as stated in the exam guide and to give you a sample question that aligns with each 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: identify common performance bottlenecks such as data skew, shuffling, and disk spilling by interpreting stage-level metrics in the Spark UI.*

A data engineer notices a batch job's duration has doubled after a new data source was onboarded. In the Spark UI, the longest stage shows that most tasks finish in under 30 seconds, but one task takes over 10 minutes. The stage's task summary shows Min/Median shuffle read near 400 MB, while Max shuffle read exceeds 5 GB.

Which solution reduces the job runtime?

A. Increase cluster size to add more executors so the slow task finishes faster

B. Confirm adaptive query execution with skew join handling is active to automatically split the oversized partition at runtime
C. Reduce **`spark.sql.shuffle.partitions`** to coalesce more work into fewer tasks
D. Manually repartition the dataset using a salt key before the join to distribute skewed keys evenly

**Question 2**
*Objective: Ingest semi-structured and unstructured data (e.g. JSON and nested data) via Lakeflow Connect and managed connectors into Delta tables*.

A data engineer needs to configure a Lakeflow Connect managed connector to ingest JSON records from a SaaS source into a Delta table in Unity Catalog. The source payload contains a nested object field called '**`order_details`**' with variable sub-keys that change between ingestion runs. The engineer wants the connector to automatically expand new sub-keys as additional columns in the target Delta table without requiring manual schema updates.

Which configuration meets the requirement?

A. Set the schema evolution mode to '**`rescue`**' in the connector's ingestion settings so that records containing unrecognized sub-keys are written to a separate rescued-data column rather than expanding the target schema.
B. Enable the '**`mergeSchema`**' option on the target Delta table's **`TBLPROPERTIES`** so that the connector merges incoming schema changes at write time without any connector-level configuration.
C. Set the schema evolution mode to '**`failOnNewColumns`**' in the connector's ingestion settings and manually run **`ALTER TABLE ADD COLUMN`** after each ingestion run to incorporate new sub-keys.
D. Set the schema evolution mode to '**`addNewColumns`**' in the connector's ingestion settings so that newly discovered sub-keys in the nested JSON field are automatically promoted to top-level columns in the target Delta table.

**Question 3**
*Objective: Prioritize between Auto Loader, Lakeflow Connect partner connectors, and other ingestion methods based on volume, frequency, data type, and governance needs*.

A data engineer is building downstream pipelines to consume Databricks audit logs from a customer-owned S3 bucket. Before implementing schema inference and checkpointing, they want to understand the delivery format, typical ingestion latency, and whether files may be overwritten.

What is Databricks audit log storage behavior?

A. Files are delivered as JSON with typical event logging under 15 minutes after delivery begins, and new deliveries can overwrite existing files
B. Files are delivered as CSV with sub-minute latency guarantees, and overwrites never occur once a file is written to preserve immutability
C. Files are delivered as Parquet with eventual consistency beyond 24 hours, and overwrites are disabled to simplify streaming ingestion
D. Files are delivered as JSON with delivery on a weekly batch cadence, and overwrites replace prior content completely without appending

**Question 4**
*Objective: Identify trends in job performance using the Lakeflow Jobs run history view.*

A data engineer enables the duration trend chart in the Lakeflow Jobs run history view for a job that executes nightly.

How does the run history view behave as a result of enabling this chart?

A. The view sets a baseline alert threshold using the average run duration across historical runs.
B. The view applies a visual indicator to rows where the run duration exceeds the average.
C. The view displays a line chart plotting the duration of each recent run over time.
D. The view generates a cost projection based on average run duration and the cluster's DBU rate.

**Question 5**
*Objective: Manage code in Databricks workspace UI using Git Folders: create/switch branches, commit, push, and open pull requests.*

A team wants a modular way to deploy, version, and orchestrate ETL pipelines in Databricks—enabling CI/CD and repeatability.

Which feature supports this requirement?

A. Use models in Unity Catalog to represent ETL jobs, where each model stores the pipeline code artifact and CI/CD promotes versions by updating model aliases tied to Job tasks.
B. Package transformation logic as wheel libraries stored in Unity Catalog Volumes and bind them to Jobs tasks to ensure deterministic deployment across environments.
C. Package API logic inside a Volume-mounted notebook, and use Jobs API v2 to trigger the notebook, depending on notebook revision history to act as a versioning system.
D. Use DABs to define resources and code assets, version them in Git, and promote deployments across environments through automated CI/CD actions.

Answers:

*1. B*

*2. D*

*3. A*

*4. C*

*5. D*