databricks

Databricks Exam Guide

Databricks Certified Data Engineer Professional

Provide Exam Guide Feedback

Please note: A new version of this exam will go live on September 30, 2025.
Please see below for the exam guide that applies to you based on your exam date.

CURRENT EXAM GUIDE

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 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. **This version covers the currently live version as of September 29th, 2025. Please check back two weeks before you take your exam to make sure you have the most current version.**

Audience Description

The Databricks Certified Data Engineer Professional certification exam assesses an individual's ability to use Databricks to perform advanced data engineering tasks. This includes an understanding of the Databricks platform and developer tools like Apache Spark, Delta Lake, MLflow, and the Databricks CLI and REST API. It also assesses the ability to build optimized and cleaned ETL pipelines. Additionally, modeling data into a Lakehouse using knowledge of general data modeling concepts will also be assessed. Finally, ensuring that data pipelines are secure, reliable, monitored, and tested before deployment will also be included in this exam. 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: 60 scored multiple-choice questions
- Time limit: 120 minutes
- Registration fee: USD 200, plus applicable taxes as required per local law
- Delivery method: Online Proctored
- Test aides: none allowed.
- Prerequisite: None required; course attendance and 1 year of hands-on experience in Databricks is highly recommended
- Validity: 2 years
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the full exam that is currently live. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam again.
- 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):
    - Databricks Streaming and LakeFlow Declarative Pipelines nm
    - Databricks Data Privacy
    - Databricks Performance Optimization
    - Automated Deployment with Databricks Asset Bundle

Exam outline

Section 1: Databricks Tooling

- Explain how Delta Lake uses the transaction log and cloud object storage to guarantee atomicity and durability
- Describe how Delta Lake's Optimistic Concurrency Control provides isolation, and which transactions might conflict
- Describe basic functionality of Delta clone.
- Apply common Delta Lake indexing optimizations including partitioning, zorder, bloom filters, and file sizes
- Implement Delta tables optimized for Databricks SQL service
- Contrast different strategies for partitioning data (e.g. identify proper partitioning columns to use)

Section 2: Data Processing (Batch processing, Incremental processing, and Optimization)

- Describe and distinguish partition hints: coalesce, repartition, repartition by range, and rebalance
- Contrast different strategies for partitioning data (e.g. identify proper partitioning columns to use)
- Articulate how to write Pyspark dataframes to disk while manually controlling the size of individual part-files.
- Articulate multiple strategies for updating 1+ records in a spark table (Type 1)
- Implement common design patterns unlocked by Structured Streaming and Delta Lake.
- Explore and tune state information using stream-static joins and Delta Lake
- Implement stream-static joins
- Implement necessary logic for deduplication using Spark Structured Streaming
- Enable CDF on Delta Lake tables and re-design data processing steps to process CDC output instead of incremental feed from normal Structured Streaming read
- Leverage CDF to easily propagate deletes
- Demonstrate how proper partitioning of data allows for simple archiving or deletion of data
- Articulate, how "smalls" (tiny files, scanning overhead, over partitioning, etc) induce performance problems into Spark queries

Section 3: Data Modeling

- Describe the objective of data transformations during promotion from bronze to silver
- Discuss how Change Data Feed (CDF) addresses past difficulties propagating updates and deletes within Lakehouse architecture
- Apply Delta Lake clone to learn how shallow and deep clone interact with source/target tables.

- Design a multiplex bronze table to avoid common pitfalls when trying to productionalize streaming workloads.
- Implement best practices when streaming data from multiplex bronze tables.
- Apply incremental processing, quality enforcement, and deduplication to process data from bronze to silver
- Make informed decisions about how to enforce data quality based on strengths and limitations of various approaches in Delta Lake
- Implement tables avoiding issues caused by lack of foreign key constraints
- Add constraints to Delta Lake tables to prevent bad data from being written
- Implement lookup tables and describe the trade-offs for normalized data models
- Diagram architectures and operations necessary to implement various Slowly Changing Dimension tables using Delta Lake with streaming and batch workloads.
- Implement SCD Type 0, 1, and 2 tables

Section 4: Security & Governance
- Create Dynamic views to perform data masking
- Use dynamic views to control access to rows and columns

Section 5: Monitoring & Logging
- Describe the elements in the Spark UI to aid in performance analysis, application debugging, and tuning of Spark applications.
- Inspect event timelines and metrics for stages and jobs performed on a cluster
- Draw conclusions from information presented in the Spark UI, Ganglia UI, and the Cluster UI to assess performance problems and debug failing applications.
- Design systems that control for cost and latency SLAs for production streaming jobs.
- Deploy and monitor streaming and batch jobs

Section 6: Testing & Deployment
- Adapt a notebook dependency pattern to use Python file dependencies
- Adapt Python code maintained as Wheels to direct imports using relative paths
- Repair and rerun failed jobs
- Create Jobs based on common use cases and patterns
- Create a multi-task job with multiple dependencies
- Design systems that control for cost and latency SLAs for production streaming jobs.
- Configure the Databricks CLI and execute basic commands to interact with the workspace and clusters.
- Execute commands from the CLI to deploy and monitor Databricks jobs.
- Use REST API to clone a job, trigger a run, and export the run output

NEW EXAM GUIDE
Use this version of the exam guide if you are taking your exam on or AFTER September 30th

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 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. **This version covers the currently live exam as of September 30th, 2025.**

Audience Description

The Databricks Certified Data Engineering Professional exam validates a candidates advanced skills In building, optimising and maintaining production-grade data engineering solutions on the Databricks Lakehouse Platform. Successful candidates demonstrate expertise across core platform features such as Delta Lake, Unity Catalog, Auto Loader, Lakeflow Definitive Pipelines, Databricks Compute (including serverless) Workflows 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.

Professionals who earn this certification are proven to have the knowledge & hands-on experience required to deliver production- ready data engineering solutions on Databricks with one or more years of experiance on the Lakehouse Platform strongly recommended.

About the Exam

- Number of items: 60 scored multiple-choice questions
- Time limit: 120 minutes
- Registration fee: USD 200, plus applicable taxes as required per local law
- Delivery method: Online Proctored
- Test aides: none allowed.
- Prerequisite: None required; course attendance and 1 year of hands-on experience in Databricks is highly recommended
- Validity: 2 years
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the full exam that is currently live. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam again.
- 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):
  - Databricks Streaming and LakeFlow Declarative Pipelines nm
  - Databricks Data Privacy
  - Databricks Performance Optimization
  - Automated Deployment with Databricks Asset Bundle

Exam outline

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

1.a:Using Python and Tools for development

- Design and implement a scalable Python project structure optimized for 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 UDFs

1.b: Building and Testing an ETL pipeline with Lakeflow 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 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 APPLY CHANGES APIs to simplify CDC in Lakeflow Declarative Pipelines
- Compare Spark Structured Streaming and Lakeflow 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, foreach, 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 built-in debugger

Section 2: Data Ingestion & Acquisition:

2.1: 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.

2.2: Create an append-only data pipeline, capable of handling both batch and streaming data using Delta

Section 3: Data Transformation, Cleansing and Quality

3.1: Write efficient Spark SQL and PySpark code to apply advanced data transformations, including window functions, joins, and aggregations, to manipulate and analyze large Datasets.

3.2: Develop a quarantining process for bad data with Lakeflow Declarative Pipelines or autoloader in classic jobs

Section 4: Data Sharing and Federation

4.1: Demonstrate delta sharing securely between Databricks deployments using Databricks

to Databricks Sharing(D2D) or to external platforms using open sharing protocol(D2O)

4.2: Configure Lakehouse Federation with proper governance across supported source Systems.

4.3: Use Delta Share to share live data from Lakehouse to any computing platform, securely

Section 5: Monitoring and Alerting

5.a: 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 inc
- Use Lakeflow Declarative Pipelines Event Logs to monitor pipelines

5.b: Alerting

- Use SQL Alerts to monitor data quality
- Use the Workflows UI and Jobs API to set up job status and performance issue notifications

Section 6:Cost & Performance Optimisation

6.1: Understand how / why using Unity Catalog managed tables reduces operation overhead and maintenance burden

6.2: Understand delta optimization techniques, such as deletion vectors and liquid clustering.

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

6.4:Apply Change Data Feed (CDF) to address specific limitations of streaming tables and enhance latency.

6.5: Use query profile to analyze the query and identify bottlenecks such as bad data skipping, inefficient types of joins, data shuffling

Section 7: Ensuring Data Security and Compliance

7.a: Applying Data Security mechanisms

- Use ACLs to secure Workspace Objects, enforcing 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

7.b: 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

8.a: Create and add descriptions/metadata about enterprise data to make it more

discoverable

8.b: Demonstrate understanding of Unity Catalog permission inheritance model

Section 9: Debugging and Deploying

9.a: 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 Declarative Pipelines event logs & the Spark UI to debugLakeflow Declarative Pipelines and Spark pipelines

9.b: Deploying CI/CD

- Build and Deploy Databricks resources using Databricks Asset Bundles
- Configure and integrate with Git-based CI/CD workflows using Databricks Git Folders for notebook and code deployment

Section 10: Data Modelling

10.1 Design and implement scalable data models using Delta Lake to manage large datasets.

10.2: Simplify data layout decisions and optimize query performance using Liquid Clustering

10.3: Identify the benefits of using liquid Clustering over Partitioning and ZOrder

10.4: 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 objectives as they are stated on the exam guide, and give you a sample question that aligns to 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: Identify the results of running a command on a Delta Lake table created with a query

A Delta Lake table was created with the query:

```
CREATE TABLE dev.my_table
USING DELTA
LOCATION "/mnt/dev/my_table"
```

Realizing that the table needs to be used by other and its name is misleading, the below code was executed:

```
ALTER TABLE dev.my_table RENAME TO dev.our_table
```

Which result will occur after running the second command?

A. The table name change is recorded in the Delta transaction log.
B. The table reference in the metastore is updated and all data files are moved.
C. The table reference in the metastore is updated and no data is changed.
D. A new Delta transaction log is created for the renamed table.
E. All related files and metadata are dropped and recreated in a single ACID transaction.

Question 2
Objective: Deduplicate data against previously processed records as it is inserted into Delta table.

A data engineer is developing an ETL workflow that could see late-arriving, duplicate records from its single source. The data engineer knows that they can deduplicate the records within the batch, but they are looking for another solution.

Which approach allows the data engineer to deduplicate data against previously processed records as it is inserted into a Delta table?

A. **VACUUM** the Delta table after each batch completes.
B. Rely on Delta Lake schema enforcement to prevent duplicate records.
C. Set the configuration **delta.deduplicate = true**.
D. Perform a full outer join on a unique key and overwrite existing data.
E. Perform an insert-only merge with a matching condition on a unique key.

Question 3
Objective: Identify how to configure all tables in the Lakehouse as external, unmanaged Delta Lake tables

The data architect has mandated that all tables in the Lakehouse should be configured as external, unmanaged Delta Lake tables.

Which approach will ensure that this requirement is met?

A. Whenever a table is being created, make sure that the **LOCATION** keyword is used.
B. When tables are created, make sure that the **EXTERNAL** keyword is used in the **CREATE TABLE** statement.
C. When the workspace is being configured, make sure that external cloud object storage has been mounted.
D. Whenever a database is being created, make sure that the **LOCATION** keyword is used.
E. Whenever a table is being created, make sure that the **LOCATION** and **UNMANAGED** keywords are used.

Question 4
Objective: Describe permission controls for Databricks jobs

A data engineering team is trying to transfer ownership from its Databricks Workflows away from an individual that has switched teams. However, they are unsure how permission controls

specifically for Databricks Jobs work.

Which statement correctly describes permission controls for Databricks Jobs?

A. The creator of a Databricks Job will always have "Owner" privileges; this configuration cannot be changed.
B. Databricks Jobs must have exactly one owner; "Owner" privileges cannot be assigned to a group.
C. Other than the default "admins" group, only individual users can be granted privileges on Jobs.
D. Only workspace administrators can grant "Owner" privileges to a group.
E. A user can only transfer Job ownership to a group if they are also a member of that group.

Question 5
Objective: Identify methods for installing python packages...
A data engineer needs to use a Python package to process data. As a result, they need to install the Python package on all of the nodes in the currently active cluster.

What describes a method of installing a Python package scoped at the notebook level to all nodes in the currently active cluster?

A. Use **%pip install** in a notebook cell
B. Use **%sh pip install** in a notebook cell
C. Run **source env/bin/activate** in a notebook setup script
D. Install libraries from PyPI using the cluster UI
E. Use **b** in a notebook cell

Answers
Question 1: C
Question 2: E
Question 3: A
Question 4: B
Question 5: A