databricks

Databricks Exam Guide

Databricks Certified Data Engineer Associate

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 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 January 1, 2024. 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 Associate certification exam assesses an individual's ability to use the Databricks Lakehouse Platform to complete introductory data engineering tasks. This includes an understanding of the Lakehouse Platform and its workspace, its architecture, and its capabilities. It also assesses the ability to perform multi-hop architecture ETL tasks using Apache Spark SQL and Python in both batch and incrementally processed paradigms. Finally, the exam assesses the tester's ability to put basic ETL pipelines and Databricks SQL queries and dashboards into production while maintaining entity permissions. Individuals who pass this certification exam can be expected to complete basic data engineering tasks using Databricks and its associated tools.

About the Exam

- Number of items: 45 multiple-choice questions
- Time limit: 90 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 six months of hands-on experience in Databricks is highly recommended
- Validity: 2 years
- Recertification: Recertification is required to maintain your certification status. To recertify, you must take the full 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. Additional time is factored into account for this content.

Recommended Training

- Instructor-led: Data Engineering with Databricks
- Self-paced: Data Engineering with Databricks (available in Databricks Academy)

Exam outline

Section 1: Databricks Lakehouse Platform

- Describe the relationship between the data lakehouse and the data warehouse.
- Identify the improvement in data quality in the data lakehouse over the data lake.
- Compare and contrast silver and gold tables, which workloads will use a bronze table as a source, which workloads will use a gold table as a source.
- Identify elements of the Databricks Platform Architecture, such as what is located in the data plane versus the control plane and what resides in the customer's cloud account
- Differentiate between all-purpose clusters and jobs clusters.
- Identify how cluster software is versioned using the Databricks Runtime.
- Identify how clusters can be filtered to view those that are accessible by the user.
- Describe how clusters are terminated and the impact of terminating a cluster.
- Identify a scenario in which restarting the cluster will be useful.
- Describe how to use multiple languages within the same notebook.
- Identify how to run one notebook from within another notebook.
- Identify how notebooks can be shared with others.
- Describe how Databricks Repos enables CI/CD workflows in Databricks.
- Identify Git operations available via Databricks Repos.
- Identify limitations in Databricks Notebooks version control functionality relative to Repos.

Section 2: ELT with Apache Spark

- Extract data from a single file and from a directory of files
- Identify the prefix included after the FROM keyword as the data type.
- Create a view, a temporary view, and a CTE as a reference to a file
- Identify that tables from external sources are not Delta Lake tables.
- Create a table from a JDBC connection and from an external CSV file
- Identify how the count_if function and the count where x is null can be used
- Identify how the count(row) skips NULL values.
- Deduplicate rows from an existing Delta Lake table.
- Create a new table from an existing table while removing duplicate rows.
- Deduplicate a row based on specific columns.
- Validate that the primary key is unique across all rows.

- Validate that a field is associated with just one unique value in another field.
- Validate that a value is not present in a specific field.
- Cast a column to a timestamp.
- Extract calendar data from a timestamp.
- Extract a specific pattern from an existing string column.
- Utilize the dot syntax to extract nested data fields.
- Identify the benefits of using array functions.
- Parse JSON strings into structs.
- Identify which result will be returned based on a join query.
- Identify a scenario to use the explode function versus the flatten function
- Identify the PIVOT clause as a way to convert data from wide format to a long format.
- Define a SQL UDF.
- Identify the location of a function.
- Describe the security model for sharing SQL UDFs.
- Use CASE/WHEN in SQL code.
- Leverage CASE/WHEN for custom control flow.

Section 3: Incremental Data Processing

- Identify where Delta Lake provides ACID transactions
- Identify the benefits of ACID transactions.
- Identify whether a transaction is ACID-compliant.
- Compare and contrast data and metadata.
- Compare and contrast managed and external tables.
- Identify a scenario to use an external table.
- Create a managed table.
- Identify the location of a table.
- Inspect the directory structure of Delta Lake files.
- Identify who has written previous versions of a table.
- Review a history of table transactions.
- Roll back a table to a previous version.
- Identify that a table can be rolled back to a previous version.
- Query a specific version of a table.
- Identify why Zordering is beneficial to Delta Lake tables.
- Identify how vacuum commits deletes.
- Identify the kind of files Optimize compacts.
- Identify CTAS as a solution.
- Create a generated column.
- Add a table comment.
- Use CREATE OR REPLACE TABLE and INSERT OVERWRITE
- Compare and contrast CREATE OR REPLACE TABLE and INSERT OVERWRITE
- Identify a scenario in which MERGE should be used.

- Identify MERGE as a command to deduplicate data upon writing.
- Describe the benefits of the MERGE command.
- Identify why a COPY INTO statement is not duplicating data in the target table.
- Identify a scenario in which COPY INTO should be used.
- Use COPY INTO to insert data.
- Identify the components necessary to create a new DLT pipeline.
- Identify the purpose of the target and of the notebook libraries in creating a pipeline.
- Compare and contrast triggered and continuous pipelines in terms of cost and latency
- Identify which source location is utilizing Auto Loader.
- Identify a scenario in which Auto Loader is beneficial.
- Identify why Auto Loader has inferred all data to be STRING from a JSON source
- Identify the default behavior of a constraint violation
- Identify the impact of ON VIOLATION DROP ROW and ON VIOLATION FAIL UPDATEfor a constraint violation
- Explain change data capture and the behavior of APPLY CHANGES INTO
- Query the events log to get metrics, perform audit loggin, examine lineage.
- Troubleshoot DLT syntax: Identify which notebook in a DLT pipeline produced an error, identify the need for LIVE in create statement, identify the need for STREAM in from clause.

Section 4: Production Pipelines

- Identify the benefits of using multiple tasks in Jobs.
- Set up a predecessor task in Jobs.
- Identify a scenario in which a predecessor task should be set up.
- Review a task's execution history.
- Identify CRON as a scheduling opportunity.
- Debug a failed task.
- Set up a retry policy in case of failure.
- Create an alert in the case of a failed task.
- Identify that an alert can be sent via email.

Section 5: Data Governance

- Identify one of the four areas of data governance.
- Compare and contrast metastores and catalogs.
- Identify Unity Catalog securables.
- Define a service principal.
- Identify the cluster security modes compatible with Unity Catalog.
- Create a UC-enabled all-purpose cluster.
- Create a DBSQL warehouse.
- Identify how to query a three-layer namespace.
- Implement data object access control
- Identify colocating metastores with a workspace as best practice.