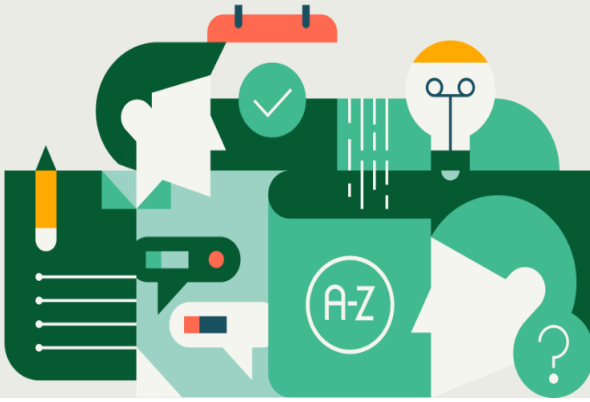
databricks

Databricks 試験ガイド

Databricks Certified Data Engineer Associate 更新試験



試験ガイドへのフィードバックを送信

この試験ガイドの目的

この試験ガイドの目的は、試験の概要と試験で扱われる内容を説明し、受験準備が整っているかどうかを判断できるようにすることです。このドキュメントは、試験に変更がある場合（およびその変更が有効になる場合）には常に更新されるため、準備に役立てられます。本ガイドは、**2026** 年 **6** 月 **15** 日時点の試験バージョンを対象としています。

対象者の説明

この Databricks Certified Data Engineer Associate 更新試験では、Databricks Data Intelligence Platform を活用して基礎的なデータエンジニアリングタスクを実行する能力を評価します。この試験では、Databricks Certified Data Engineer Associate として、データの取り込み、データの読み込み、データ変換、モデリングに関連するタスクの知識を評価します。具体的には、PySpark / SQL を使用した抽出、変換、読み込み（ETL）タスクの実行、Lakeflow Jobs の利用、CI/CD などです。さらにこの試験では、受験者のトラブルシューティング、監視、最適化の手法に関する理解と、Databricks Platform 内でのガバナンスとセキュリティの実現に関する知識も評価します。

試験について

- 採点対象の項目数: 採点対象となる多肢選択問題 25 問。
- 制限時間: 60 分。
- 受験料: 200 USD
- 配信方法: オンライン監督付きのみ
- 試験補助: 使用不可。
- 前提条件: この更新試験は、既存の Databricks Certified Data Engineer Associate 認定資格を保有しており、その資格の有効期限が切れているか、有効期限まで 2 か月以内である個人が受験できます。受験者は、推奨されるトレーニングを修了し、Databricks Platform での実務経験を少なくとも 12 か月有していることを強く推奨します。
- 有効期間: 2 年間。
- 再認定: 受験者は、有効な試験またはその年に提供される特定の再認定バージョンに合格することで、2 年ごとに認定資格を更新する必要があります。試験の準備については、試験の Web ページの「Getting Ready for the Exam」セクションを確認してください。

推奨トレーニング

- インストラクター主導: Data Engineering with Databricks
- セルフペース（Databricks Academy で利用可能）:
  - 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

試験の構成

セクション **1:** データの取り込みと読み込み（**32%**）

- COPY INTO を使用して、クラウドオブジェクトストレージから Unity Catalog ガバナンス下の Delta テーブルへの増分的なファイル取り込みを実行する。
- Auto Loader を使用して、Unity Catalog ガバナンス下の Delta テーブルにファイルを増分的に取り込む。
- 多様なエンタープライズソースから Unity Catalog ガバナンス下のテーブルへ、確実にデータを取り込むように Lakeflow Connect を構成する。
- ノートブック内で JDBC/ODBC または REST クライアントを使用して、データをクラウドストレージ、または直接 Unity Catalog ガバナンス下のテーブルに配置する。
- ボリューム、頻度、データ型、ガバナンスのニーズに基づいて、Auto Loader、Lakeflow Connect パートナーコネクタ、およびその他の取り込み方法の間で優先順位を付ける。
- 半構造化データおよび非構造化データ（JSON やネストされたデータなど）を、Lakeflow Connect とマネージドコネクタを介して Delta テーブルに取り込む。

セクション **2:** データ変換とモデリング（**24%**）

- PySpark/SQL で bronze テーブルを読み取ってデータクレンジングを適用し（null の処理、データ型の標準化）、その結果を silver テーブルに書き込む。
- 一般的な Spark のチューニングパラメーターを特定し、それらがパフォーマンスに与える一般的な影響を理解する。
- Unity Catalog の Gold レイヤーのオブジェクト（マテリアライズドビュー、ビュー、ストリーミングテーブル、通常のテーブル）と、BI/分析でのそれらの用途を区別する。
- 信頼性の高い Silver および Gold データセットを確保するために、データ品質チェックと検証ルールを適用する。

セクション **3: Lakeflow Jobs** の利用（**12%**）

- Lakeflow Jobs を使用して制御フロー（タスクの再試行、分岐、ループ）を実装し、複数ステップのデータワークフローをオーケストレーションする。
- DAG ベースのワークフローを使用して、Lakeflow Jobs で一般的なタスクタイプと依存関係を構成する。

- スケジュール、ファイル到着、テーブル更新、継続的トリガーを含む、Lakeflow Jobs のスケジュールとトリガーを構成する。

セクション **4: CI/CD** の実装（**12%**）

- Git フォルダーを使用して Databricks ワークスペース UI でコードを管理する（ブランチの作成/切り替え、コミット、プッシュ、プルリクエストの作成）。
- Declarative Automation Bundle の環境ターゲット（dev、test、prod）を使用して、コードベースを環境間で昇格させる。
- Databricks CLI を使用して、Declarative Automation Bundle（DAB）を検証してデプロイする。

セクション **5:** トラブルシューティング、監視、最適化（**12%**）

- Lakeflow Jobs の実行履歴ビューを使用して、ジョブのパフォーマンスの傾向を特定する。
- Lakeflow Jobs UI を使用して、パイプラインの健全性、ジョブのステータス、DAG タスクグラフ、実行時間、失敗率を監視する。
- マネージドの Delta Lake 機能としての Liquid Clustering と予測的最適化の目的と利点を特定する。

セクション **6:** ガバナンスとセキュリティ（**8%**）

- 列レベルのマスキングと行レベルのセキュリティの目的を理解し、それらをいつ適用すべきかを特定する。
- Unity Catalog の ABAC ポリシーの機能と、それらが一元化されたデータアクセス制御をどのようにサポートするかを理解する。

サンプル問題

これらの問題は、以前のバージョンの試験から引退したものです。その目的は、試験ガイドに記載されている目標を示し、各目標に沿ったサンプル問題を提供することです。試験ガイドには、試験で扱われる可能性のある目標が記載されています。認定試験の準備をする最善の方法は、試験ガイドの試験の構成を確認することです。

問題 **1**
目標: *Spark UI* のステージレベルのメトリクスを解釈して、データスキュー、シャッフル、ディスクスピルなどの一般的なパフォーマンスのボトルネックを特定する。

データエンジニアが、新しいデータソースを追加した後、あるバッチジョブの所要時間が 2 倍になったことに気づきます。Spark UI では、最も長いステージにおいて、ほとんどのタスクが 30 秒未満で完了する一方、1 つのタスクは 10 分以上かかっています。そのステージのタスクサマリーでは、シャッフル読み取りの最小値/中央値が 400 MB 前後であるのに対し、最大値は 5 GB を超えています。

ジョブの実行時間を短縮するソリューションはどれですか？

A. 遅いタスクをより速く完了させるために、クラスターのサイズを増やしてエグゼキューターを追加する
B. 実行時に過大なパーティションを自動的に分割するために、スキュージョイン処理を備えた Adaptive Query Execution が有効になっていることを確認する
C. より多くの処理を少数のタスクにまとめるために、**`spark.sql.shuffle.partitions`** を減らす
D. スキューしたキーを均等に分散させるために、結合の前にソルトキーを使用してデータセットを手動で再パーティション化する

問題 **2**
目標: 半構造化データおよび非構造化データ（*JSON* やネストされたデータなど）を、*Lakeflow Connect* とマネージドコネクタを介して *Delta* テーブルに取り込む。

データエンジニアが、SaaS ソースから JSON レコードを Unity Catalog 内の Delta テーブルに取り込むために、Lakeflow Connect のマネージドコネクタを構成する必要があります。ソースのペイロードには、取り込み実行ごとに変化する可変のサブキーを持つ '**`order_details`**' という名前のネストされたオブジェクトフィールドが含まれています。エンジニアは、手動でのスキーマ更新を必要とせずに、コネクタが新しいサブキーをターゲットの Delta テーブルの追加の列として自動的に展開することを望んでいます。

要件を満たす構成はどれですか？

A. コネクタの取り込み設定でスキーマ進化モードを '**`rescue`**' に設定し、認識されないサブキーを含むレコードが、ターゲットスキーマを拡張するのではなく、別個の rescued-data 列に書き込まれるようにする。
B. ターゲットの Delta テーブルの **`TBLPROPERTIES`** で '**`mergeSchema`**' オプションを有効にし、コネクタレベルの構成なしに、コネクタが書き込み時に受信するスキーマ変更をマージするようにする。
C. コネクタの取り込み設定でスキーマ進化モードを '**`failOnNewColumns`**' に設定し、新しいサブキーを組み込むために、取り込み実行のたびに **`ALTER TABLE ADD COLUMN`** を手動で実行する。
D. コネクタの取り込み設定でスキーマ進化モードを '**`addNewColumns`**' に設定し、ネストされた JSON フィールドで新しく検出されたサブキーが、ターゲットの Delta テーブルの最上位の列として自動的に昇格されるようにする。

問題 **3**
目標: ボリューム、頻度、データ型、ガバナンスのニーズに基づいて、*Auto Loader*、*Lakeflow Connect* パートナーコネクタ、およびその他の取り込み方法の間で優先順位を付ける。

データエンジニアが、顧客所有の S3 バケットから Databricks 監査ログを利用する下流のパイプラインを構築しています。スキーマ推論とチェックポイントを実装する前に、配信形式、一般的な取り込みレイテンシ、およびファイルが上書きされる可能性があるかどうかを理解したいと考えています。

Databricks 監査ログのストレージの動作はどのようなものですか？

A. ファイルは JSON として配信され、通常、配信開始後 15 分以内にイベントがログに記録され、新しい配信によって既存のファイルが上書きされる可能性がある

B. ファイルは CSV として配信され、1 分未満のレイテンシが保証され、不変性を保つために、ファイルが一度書き込まれると上書きが発生することはない
C. ファイルは Parquet として配信され、24 時間を超える結果整合性を持ち、ストリーミングの取り込みを簡素化するために上書きが無効化されている
D. ファイルは JSON として配信され、毎週のバッチ間隔で配信され、上書きは追加せずに以前の内容を完全に置き換える

問題 **4**
目標: *Lakeflow Jobs* の実行履歴ビューを使用して、ジョブのパフォーマンスの傾向を特定する。

データエンジニアが、毎晩実行されるジョブについて、Lakeflow Jobs の実行履歴ビューで所要時間の傾向チャートを有効にします。

このチャートを有効にした結果、実行履歴ビューはどのように動作しますか？

A. ビューは、過去の実行全体の平均所要時間を使用して、ベースラインのアラートしきい値を設定する。
B. ビューは、実行の所要時間が平均を超える行に視覚的なインジケーターを適用する。
C. ビューは、最近の各実行の所要時間を時系列でプロットした折れ線グラフを表示する。
D. ビューは、平均所要時間とクラスターの DBU レートに基づいてコスト予測を生成する。

問題 **5**
目標: *Git* フォルダーを使用して *Databricks* ワークスペース *UI* でコードを管理する（ブランチの作成/切り替え、コミット、プッシュ、プルリクエストの作成）。

あるチームが、Databricks で ETL パイプラインをデプロイ、バージョン管理、オーケストレーションするためのモジュール式の方法を求めており、CI/CD と再現性を実現したいと考えています。

この要件をサポートする機能はどれですか？

A. Unity Catalog のモデルを使用して ETL ジョブを表現する。各モデルはパイプラインのコードアーティファクトを格納し、CI/CD は、ジョブタスクに紐づくモデルエイリアスを更新することでバージョンを昇格させる。
B. 変換ロジックを、Unity Catalog ボリュームに格納された wheel ライブラリとしてパッケージ化し、それらをジョブタスクにバインドして、環境間で確定的なデプロイを保証する。
C. API ロジックを Volume にマウントされたノートブック内にパッケージ化し、Jobs API v2 を使用してそのノートブックをトリガーし、バージョン管理システムとしてノートブックのリビジョン履歴に依存する。
D. DAB を使用してリソースとコードアセットを定義し、それらを Git でバージョン管理し、自動化された CI/CD アクションを通じて環境間でデプロイを昇格させる。

解答:

1. B
2. D
3. A
4. C
5. D