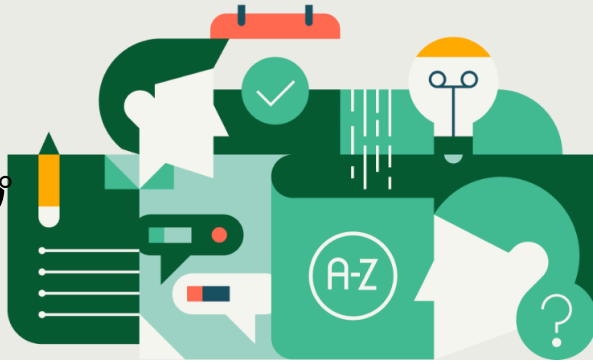
databricks

Databricks 試験ガイド

Databricks 認定データエンジニアプロフェッショナル



試験ガイドに関するフィードバックの提供

この試験ガイドの目的

この試験ガイドでは、試験の準備に役立てていただくために、試験の概要と試験の対象範囲について説明します。試験に何らかの変更がある場合には(そして、それらの変更が試験に反映される際には)、試験の準備を行えるように、このドキュメントは随時更新されます。このバージョンは、**2025 年 9 月 30 日現在の実施試験に対応しています**。試験を受ける **2** 週間前に、このドキュメントが最新版であることを再度ご確認ください。

対象者についての説明

Databricks Certified Data Engineering Professional試験では、Databricks Lakehouse Platform上で本番グレードのデータエンジニアリングソリューションを構築、最適化、維持するための受験者の高度なスキルを検証します。合格者は、Delta Lake, Unity Catalog, Auto Loader, Lakeflow Declarative Pipelines、Databricks Compute (serverless) Lakeflow Jobs、Medallion Architectureなどのコアプラットフォーム機能に関する専門知識を実証します。この認定資格では、安全で信頼性が高く、費用対効果の高い ETL パイプラインを設計し、Python と SQL を使用してさまざまなソースからの複雑なデータを処理し、スキーマ管理、可観測性、ガバナンス、パフォーマンスの最適化におけるベストプラクティスを適用する能力を評価します。受験者は、ストリーミングワークロードの実装、Workflowsのオーケストレーション、DevOps と CI/CD の活用、Databricks CLI、 REST API やアセットバンドルなどのツールを使用したデプロイについてもテストされます。この認定資格を取得したプロフェッショナルは、Databricks で本番環境に対応したデータエンジニアリング ソリューションを提供するために必要な知識と実践的な経験を持っていることが証明されており、Lakehouse プラットフォームで 1 年以上の経験があることを強くお勧めします。

試験について

- 問題数:採点対象の多肢選択問題59問
- 制限時間:120分
- 登録料: 200 米ドル、および現地の法律によって適用される税金が加算されます
- 実施方法:オンライン監督付き
- テストエイド: 許可されていません。
- 前提条件: 必要なし。Databricksでのコース受講と1年間の実務経験を強くお勧めします
- 有効期間:2年
- 再認定: 認定ステータスを維持するには、2 年ごとに再認定が必要です。再認定を受けるには、現在実施中の完全な試験を受ける必要があります。試験のウェブページの「試験の準備」セクションを確認して、再度試験を受ける準備をしてください。

- 採点対象外の内容: 今後使用する統計情報を収集するために、試験には採点対象外の項目が含まれている場合があります。これらの項目はフォーム上では識別されず、スコアには影響しません。この内容については、追加の時間が考慮されます。

推奨されるトレーニング

- インストラクター主導の上級者向け Data Engineering With Databricks
- セルフペース (Databricks Academy で利用可能):
    - Databricks Streaming および LakeFlow Declarative Pipelines
    - Databricks Data Privacy
    - Databricks Performance Optimization
    - Automated Deployment with Databricks Asset Bundle

試験の概要

セクション **1: Python** と **SQL** を使用したデータ処理用コードの開発

- 開発に Python とツールを使用する
    - Databricks アセットバンドル (DAB) に最適化されたスケーラブルな Python プロジェクト構造を設計および実装し、モジュール開発、デプロイの自動化、CI/CD　統合を可能にします。
    - PyPI パッケージ、ローカルWheel、ソースアーカイブを含む、Databricks での外部サードパーティライブラリのインストールと依存関係を管理およびトラブルシューティングします。
    - Pandas/Python UDF を使用してユーザー定義関数 (UDF) の開発
- Databricks platform で Lakeflow Declarative Pipelines、SQL、Apache Spark を使用した ETL パイプラインの構築とテスト
    - Lakeflow Declarative Pipelines と Autoloader を使用して、バッチおよびストリーミングデータ用の信頼性の高い本番環境対応のデータパイプラインを構築および管理します。
    - UI/APIs/CLI でジョブを使用して ETL ワークロードを作成および自動化します。
    - マテリアライズドビューと比較したストリーミングテーブルの長所と短所を説明する。
    - APPLY CHANGES APIs を使用して、Lakeflow Declarative Pipelines 内の CDC を簡略化します
    - Spark Structured Streaming と Lakeflow 宣言型パイプラインを比較して、スケーラブルな ETL パイプラインを構築するための最適なアプローチを決定します。
    - 制御フロー演算子 (if/else や foreach など) を使用するパイプライン コンポーネントを作成します。
    - 環境と依存関係に適した構成、ノートブック　タスク用の高メモリ、再試行を禁止する自動最適化を選択します。
    - assertDataFrameEqual、assertSchemaEqual、DataFrame.transform、およびテストフレームワークを使用して単体テストと統合テストを開発し、組み込みのデバッガを含むコードの正確性を確保します。

セクション **2:** データの取り込みと取得**:**

- メッセージバスやクラウドストレージなどのさまざまなソースから、Delta　Lake,　Parquet,　ORC, Avro, JSON, CSV, XML、TEXT、バイナリなどのさまざまなデータ形式を効率的に取り込むためのデータ取り込みパイプラインを設計および実装します。

- Delta を使用して、batch と streaming の両方のデータを処理できる追加専用データパイプラインを作成します。

セクション **3:** データの変換、クレンジング、および品質

- 効率的な Spark SQL および PySpark コードを記述して、ウィンドウ関数、結合、集計などの高度なデータ変換を適用し、大規模なデータセットを操作および分析します。
- クラシック ジョブで LakeFlow Declarative Pipelines または Autoloader を使用して、不良データの検疫プロセスを開発します。

セクション **4:** データ共有とフェデレーション

- Databricks to Databricks Sharing(D2D)を使用したDatabricks環境間、またはオープン共有プロトコル(D2O)を使用した外部プラットフォーム間で、Delta Sharingを安全に実証します。
- サポートされているソースシステム全体で適切なガバナンスを使用してLakehouse Federationを構成します。
- Delta Share を使用して、Lakehouse から任意のコンピューティング プラットフォームにライブデータを共有します。

セクション **5:** モニタリングとアラート

- モニタリング
  - システムテーブルを使用して、リソース使用率、コスト、監査、ワークロードモニタリングの可観測性を確保します。
  - Query Profiler UI と Spark UI を使用してワークロードをモニタリングします。
  - Databricks REST APIs/Databricks CLI を使用してジョブとパイプラインをモニタリングします。
  - パイプラインを監視するには、LakeFlow Declarative Pipelines イベントログを使用します。
- 警告
  - SQL アラートを使用して、データ品質を監視します。
  - Workflows UI とジョブ API を使用して、ジョブの状態とパフォーマンスの問題の通知を設定します。

セクション**6:**コストとパフォーマンスの最適化

- Unity Catalog マネージドテーブルを使用すると、運用のオーバーヘッドとメンテナンスの負担が軽減される方法と理由を理解します。
- 削除ベクトルやリキッドクラスタリングなどのDelta最適化手法を理解します。
- 大規模なデータセットに対するクエリのパフォーマンスを確保するために Databricks で使用される最適化手法 (データスキップ、ファイルプルーニングなど) を理解します。
- Change Data Feed (CDF) を適用して、ストリーミングテーブルの特定の制限に対処し、レイテンシーを強化します。
- クエリープロファイルを使用してクエリーを分析し、不適切なデータ スキップ、非効率的な種類の結合、データ シャッフルなどのボトルネックを特定します。

セクション**7:**データセキュリティの確保とコンプライアンス

- データセキュリティメカニズムの適用。

        - ACL を使用して workspace オブジェクトを保護し、最小特権やポリシー適用などの原則を適用します。
        - 行フィルターと列マスクを使用して、機密性の高いテーブル データをフィルター処理およびマスクします。
        - 機密データにハッシュ化、トークン化、抑制、汎化などの匿名化と仮名化の方法を適用します。
- コンプライアンスの確保
    - PII のマスキングを検出して適用するコンプライアンスのバッチ & streaming pipeline を実装し、データのプライバシーを確保します。
    - データ保持ポリシーでコンプライアンスを確保するデータパージソリューションを開発します。

セクション **8:** データガバナンス
- エンタープライズデータに関する説明/メタデータを作成して追加し、ITを見つけやすくします。
- Unity Catalog権限継承モデルの理解を示します。

セクション **9:** デバッグとデプロイ
- デバッグとトラブルシューティング
    - Spark UI、クラスターログ、システム テーブルとクエリー プロファイルを使用して、関連する診断情報を特定し、エラーのトラブルシューティングを行います。
    - エラーを分析し、ジョブの修復とパラメーター オーバーライドを使用して失敗したジョブ実行を修復します。
    - LakeFlow Declarative Pipelines イベント ログと Spark UI を使用して LakeFlow をデバッグします
    - 宣言型パイプラインと Spark パイプライン。
- CI/CD のデプロイ
    - Databricks アセットバンドルを使用して Databricks リソースをビルドおよびデプロイします。
    - notebook とコードのデプロイに Databricks Git フォルダーを使用して Git ベースの CI/CD Workflows を構成して統合します。

セクション **10:** データモデリング
- Delta Lake を使用してスケーラブルなデータモデルを設計および実装し、大規模なデータセットを管理します。
- Liquid Clusteringを使用して、データレイアウトの決定を簡素化し、クエリーパフォーマンスを最適化します。
- パーティショニングや ZOrder よりも Liquid Clustering を使用する利点を特定する。
- 分析ワークロードのディメンションモデルを設計し、効率的なクエリーと集計を保証します。

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廃止されたものです。目的は、試験ガイドに記載されている目的を示し、目的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、試験の出題対象となる目的の一覧が記載されています。認定試験の準備を行う最善の方法は、試験ガイドの試験の概要を確認することです。

質問 **1**

目標*: Delta Lake*のカタログメタストアの操作と*ACID*コンプライアンスの動作を理解する。

以下のクエリによってDelta Lakeテーブルが作成されました:

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

元のクエリーに入力ミスがあることに気づいて、次のコードが実行されました:

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

2 番目のコマンドを実行した後、どちらの結果が発生しますか?

A. 関連するすべてのファイルとメタデータが削除され、1 つの ACID トランザクションで再作成されます。
B. テーブル名の変更は、Delta トランザクション ログ に記録されます。
C. 名前が変更されたテーブルに対して、新しい Delta トランザクション ログ が作成されます。
D. メタストア内のテーブル参照が更新されます。

質問 **2**
目標*: Spark* 構造化ストリーミングの動作を理解し、本番運用の*SLA*対応パイプラインの最適なアプローチを決定します。

本番運用にデプロイされた構造化ストリーミング ジョブで、その日のピーク時に遅延が発生しています。現時点では、通常の実行中に、データの各マイクロバッチは 3 秒未満で処理されます。その日のピーク時には、各マイクロバッチの実行時間は非常に不安定になり、30秒を超えることもあります。現在、streaming 書き込みは 10 秒のトリガー間隔で構成されています。

他の変数は一切変更せずに、レコードを 10 秒未満で処理する必要があると仮定した場合、要件を満たす調整はどれですか。

A. trigger once オプションを使用して、8 秒ごとに query を実行するように Databricks ジョブを構成します。これにより、バックログされたすべてのレコードが各 batch で処理されます。
B. トリガー間隔を 5 秒に減らします。バッチをより頻繁にトリガーすると、レコードが滞留しなくなり、大きなバッチによってスピルが発生するのを防ぐことができます。
C. トリガー間隔を 5 秒に減らします。バッチをより頻繁にトリガーすると、アイドル状態のエグゼキューターは、前のバッチから実行時間の長いタスクが終了している間に、次の batch 処理を開始できます。
D. トリガー間隔は、チェックポイント・ディレクトリーを変更せずに変更することはできません。現在のストリーム状態を維持するには、シャッフル パーティションの数を増やして並列処理を最大化します。

質問 **3**
目的:匿名化、仮名化の方法を適用し、ハッシュ化、トークン化、

抑制と機密データへの汎化

データエンジニアリングチームは、数千のテーブルとビューを持つエンタープライズ システムを Lakehouse に移行しています。ターゲット アーキテクチャは、一連の ブロンズ、シルバー、およびゴールドテーブルを使用して実装することを計画しています。ブロンズテーブルはほぼ本番環境のデータエンジニアリング ワークロードのみに使用され、シルバーテーブルはデータエンジニアリングと機械学習の両方のワークロードをサポートするために使用されます。ゴールド テーブルは、主に Business Intelligence とレポート作成の目的を果たします。個人識別情報(PII)はデータのすべての層に存在しますが、シルバーレベルとゴールドレベルのすべてのデータに対して仮名化と匿名化のルールが設けられています。

この組織は、セキュリティ上の懸念を軽減しながら、多様なチーム間でのコラボレーション能力を最大化することに関心を持っています。

このシステムを実装するためのベスト プラクティスを例示する記述は次のうちどれですか。

A. データ品質層に基づいて個別のデータベースにテーブルを分離すると、データベース ACL によるアクセス許可の管理が容易になり、マネージド テーブルのデフォルト ストレージの場所を物理的に分離できます。
B. すべての運用テーブルを 1 つのデータベースに格納すると、Lakehouse全体で使用可能なすべてのデータ資産の統一されたビューが提供され、すべてのユーザーにこのデータベースに対するビュー特権を付与することで、データが容易に発見できるようにします。
C. Databricks 上のデータベースは単なる論理的な構造であるため、データベース編成に関する選択は、Lakehouseのセキュリティや検出可能性に影響を与えません。
D. デフォルト Databricks データベースでの作業は、DBFS root で作成されるため、マネージド テーブルを操作するときに最大のセキュリティを提供します。

質問 **4**
目的*: Delta Lake* を使用してスケーラブルなデータモデルを設計および実装し、大規模なデータセットを管理します。

ユーザーからのコンテンツ投稿に関するメタデータを表す Delta Lake テーブルは、次の `schema`:

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

上記のスキーマに基づいて、Delta Lake テーブルをパーティション分割するのに適した列はどれですか。

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

質問 **5**
目標*: Unity Catalog* 権限継承モデルの理解を示す

user_ltv という名前のテーブルは、さまざまなチームのデータ アナリストが使用する ビュー を作成するために使用されています。workspace のユーザーはグループに構成され、ACL を使用したデータ アクセスの設定に使用されます。

user_ltv テーブルには、次のスキーマがあります。

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

次の ビュー 定義が実行されます。

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

marketing グループのメンバーではないアナリストが、次の クエリーを実行しました:

```
SELECT * FROM email_ltv
```

この クエリー の結果はどうなりますか?

A. **email** 列と **ltv** 列のみが返され、各行のemail列には**"REDACTED"** という文字列が含まれます。
B. 3 つの列が返されますが、1 つの列には **"REDACTED"** という名前が付けられ、null 値のみが含まれます。
C. **email** 列と **ltv** 列のみが返され、email列にはすべて NULL 値が格納されます。
D. **email** と **ltv** 列は、**user_ltv**内の値で返されます。

質問6:
目的 環境と依存関係に適した構成、ノートブック タスク用の高メモリ、再試行を許可しない自動最適化を選択します。

ビジネスレポートチームは、ダッシュボードのデータを毎時更新するよう求めています。パイプラインのデータを抽出、変換、およびロードするパイプラインの合計処理時間は 10 分です。

正常な動作条件を前提とした場合、最も低いコストでSLA(Service Level Agreement)要件を満たす構成はどれですか。

A. 専用の対話型クラスターで 1 時間に 1 回パイプラインを実行するジョブをスケジュールします。
B. 新しいジョブ クラスターで 1 時間に 1 回パイプラインを実行するジョブをスケジュールします。
C. トリガー間隔が 60 分の構造化ストリーミング ジョブをスケジュールします。
D. 新しいデータが特定のディレクトリに到達するたびに実行するジョブを構成します。

質問7:
目標 *Notebook*開発環境、変数管理、安全で構成可能なコードの作成について理解します。

セキュリティチームは、Databricks シークレットモジュールを利用して外部データベースに接続できるかどうかを検討しています。

すべての Python 変数を strings で定義してコードをテストした後、Databricks シークレットモジュールにパスワードをアップロードし、現在アクティブなユーザーに適切なアクセス許可を構成します。次に、コードを次のように変更します (他のすべての変数は変更しません)。

```
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)
  )
```

上記のコードが実行されたときに何が起こるかを説明しているステートメントはどれですか。

A. 外部テーブルへの接続は成功します。文字列 **"REDACTED"** が出力されます。
B. 外部テーブルへの接続は成功します。**password** の文字列値はプレーンテキストで出力されます。
C. インタラクティブ入力ボックスがノートブックに表示されます。正しいパスワードが入力されると、接続は成功し、パスワードがプレーンテキストで出力されます。
D. インタラクティブ入力ボックスがノートブックに表示されます。正しいパスワードが入力されると、接続は成功し、エンコードされたパスワードは DBFS に保存されます。

質問 8:
目的:大規模なデータセット(データスキップ、ファイルプルーニングなど)に対するクエリのパフォーマンスを確保するために*Databricks*で使用される最適化手法を理解する

データ インジェスト タスクでは、1 TB の JSON dataset を 512 MB のターゲット パーツ ファイル サイズで Parquet に書き出す必要があります。Delta Lake ではなく Parquet が使用されているため、自動最適化や自動圧縮などの組み込みのファイル サイズ調整機能は使用できません。

データをシャッフルせずに最高のパフォーマンスが得られる戦略はどれですか?

A. データを取り込み、ナロー変換を実行し、2,048 パーティション (1TB*1024*1024/512) に再パーティション分割してから、Parquet に書き込みます。
B. **spark.sql.adaptive.advisoryPartitionSizeInBytes** を 512 MB バイトに設定し、データを取り

込み、ナロー変換を実行し、2,048 パーティション (1TB*1024*1024/512) に結合してから、Parquet に書き込みます。
C. **spark.sql.files.maxPartitionBytes** を 512 MB に設定し、データを取り込み、ナロー変換を実行してから、Parquet に書き込みます。
D. **spark.sql.shuffle.partitions** を 2,048 パーティション (1TB*1024*1024/512) に設定し、データを取り込み、ナロー変換を実行し、それを並べ替えてデータを最適化し (データを自動的に再パーティション化)、Parquet に書き込みます。

質問 9:
内容*: Delta Lake*クローンを適用して、シャロークローンとディープクローンがソース/ターゲットテーブルとどのように相互作用するかを学びます。

マーケティングチームは、集計テーブルのデータを営業組織と共有しようとしていますが、チームで使用されているフィールド名が一致せず、マーケティング固有のフィールドの多くが営業組織で承認されていません。

シンプルさを重視しながら状況に対処するソリューションはどれですか?

A. マーケティングテーブルにビューを作成し、営業チームに対して承認されたフィールドのみを選択します。エイリアスを使用して営業の命名規則に標準化する必要があるフィールドの名前を設定します。
B. 必要なスキーマで新しいテーブルを作成し、Delta Lake の DEEP CLONE 機能を使用して、1 つのテーブルにコミットされた変更を対応するテーブルに同期します。
C. CTAS ステートメントを使用して、マーケティング テーブルから派生テーブルを作成します。変更を伝播するように運用ジョブを構成します。
D. 現在の運用パイプラインに並列テーブル書き込みを追加し、必要に応じてマーケティング テーブルと異なる新しい営業テーブルを更新します。

質問 10:
目的*:* 複数の依存関係を持つマルチタスクジョブを作成する

あるDatabricks ジョブは 3 つのタスクで構成されており、それぞれが Databricks notebook です。タスク A は他のタスクに依存しません。タスク B と C は並列に実行され、それぞれがタスク A に直列依存しています。

スケジュール通りにタスク A と B が正常に完了し、タスク C が失敗した場合、結果はどうなりますか。

A. タスク A と B に関連付けられている Databricks notebook で表されるすべてのロジックが正常に完了します。タスク C の一部の操作が正常に完了した可能性があります。
B. すべてのタスクが正常に完了しない限り、変更は Lakehouse にコミットされません。タスク C が失敗したため、すべてのコミットが自動的にロールバックされます。
C. タスク A と B に関連付けられているノートブックで表されるすべてのロジックが正常に完了します。タスク C で行われた変更は、タスクの失敗によりロールバックされます。
D. すべてのタスクは依存関係グラフとして管理されるため、すべてのタスクが正常に完了するまで、Lakehouse の変更はコミットされません。

答え
質問1: D
質問2:B
質問3:A
質問4:C
質問5:A
質問6:B
質問7:A
質問8:C
質問9:A
質問10:A