databricks



Databricks Exam 가이드

Databricks Certified 데이터 엔지니어 Professional

시험 가이드 피드백 제공

정보 업데이트

업데이트	날짜
시험 공개	2026년 11월 20일
시험 가이드 업데이트	2026년 11월 20일

시험 날짜 2주 전에 당사 웹페이지의 시험 가이드를 확인하여 현재 버전의 시험을 준비하고 있는지 확인하십시오.

이 시험 가이드의 목적

이 시험 가이드의 목적은 시험에 대한 개요와 시험 범위를 제공하여 시험 준비 상태를 파악하는 데 도움을 드리는 것입니다. 이 문서는 시험에 변경 사항이 있을 때마다 업데이트되며(변경 사항의 적용 시기도 명시됩니다), 이를 통해 자신 있게 준비하실 수 있습니다. 이 가이드는 위에 명시된 발효일 기준으로 현재 운영 중인 시험 버전을 다룹니다. 최신 버전을 확인하시려면 시험 2주 전에 다시 방문해 주시기 바랍니다.

대상 독자 설명

Databricks Certified Data Engineering Professional 시험은 Databricks Platform에서 생산 수준의 데이터 엔지니어링 솔루션을 구축, 최적화 및 유지 관리하는 응시자의 고급 역량을 평가합니다. 합격자는 Databricks SQL, Unity Catalog, Auto Loader, Lakeflow Pipelines, Databricks Compute(서버리스 포함), Lakeflow Jobs, Lakehouse Federation, OpenSharing(D2D 및 Databricks-to-Open), Clean Rooms, 메달리온 아키텍처 등 핵심 플랫폼 기능 전반에 걸친 전문성을 갖추고 있음을 입증합니다. 이 시험의 합격자는 안전하고 신뢰할 수 있으며 비용 효율적인 ETL 파이프라인을 설계하고, Python과 SQL을 사용하여 다양한 소스의 복잡한 데이터를 처리하며, 스키마 관리, 관찰 가능성, 거버넌스, 규정 준수 및 성능 최적화에 관한 모범 사례를 적용할 수 있습니다. 응시자는 스트리밍 워크로드 구현, 워크플로 오케스트레이션, 캐싱, 디버깅 기법, 최적화 기법, DevOps 및 CI/CD 활용, Databricks CLI 및 REST API와 같은 도구를 사용한 배포, AI Functions 및 Declarative

Automation Bundles 적용에 대해서도 평가받습니다. 이 인증 시험의 합격자는 Databricks 및 관련 도구를 사용하여 고급 데이터 엔지니어링 작업을 수행할 수 있습니다.

시험 소개

항목	세부 정보
채점 문항 수	60 scored multiple-choice questions
채점되지 않는 문항 포함 가능	채점되지 않는 문항이 포함될 수 있습니다(아래 '채점되지 않는 콘텐츠' 참조)
응시 중 볼 수 있는 총 문항 수	채점 문항 60개 및 채점되지 않는 문항
제한 시간	120분(채점되지 않는 문항을 고려한 추가 시간이 이미 반영되어 있습니다)
등록 비용	USD 200 및 현지 법률에 따라 적용되는 세금 별도
응시 방식	온라인 감독 또는 시험 센터
제공 언어	영어, 일본어, 브라질 포르투갈어(pt-BR), 한국어
시험 보조 도구	허용되지 않음
사전 요구 사항	없음. 단, 관련 학습을 적극 권장합니다
권장 경험	제시된 시험 아웃라인에 해당하는 데이터 엔지니어링 업무에 대한 1년간의 실무 경험을 적극 권장합니다
유효 기간	2년

재인증**:** 인증된 상태를 유지하려면 **2**년마다 재인증이 필요합니다**.** 재인증을 받으려면 현재 운영 중인 시험에 응시해야 합니다**.**

채점되지 않는 콘텐츠**:** 시험에는 향후 활용을 위한 통계적 정보 수집을 목적으로 채점되지 않는 문항이 포함될 수 있습니다**.** 이러한 문항은 시험에서 별도로 표시되지 않으며 점수에 영향을 미치지 않고**,** 해당 문항을 고려한 추가 시간이 제한 시간에 반영됩니다**.**

권장 준비 사항

학습

- 강사 주도: Advanced 데이터 엔지니어링 with Databricks
- 자기 주도 학습 (Databricks Academy에서 이용 가능):
    - Apache Spark™ Declarative 파이프라인의 고급 기법
    - Databricks Data Privacy

- Databricks 성능 최적화
- Declarative Automation Bundles를 사용한 자동화 배포

AI Prep 가이드 - 모든 Databricks 시험

Databricks AI Prep 가이드는 이미 사용 중인 AI 챗봇을 Databricks 인증 시험을 위한 맞춤형 튜터로 활용하는 방법을 안내합니다. 공식 시험 가이드를 기반으로 챗봇을 준비시키고, AI가 제공하는 경향이 있는 오래된 답변을 파악하며, 본인의 취약 부분을 찾아내고, 이 모든 것을 모든 Databricks 시험에서 요구되는 실습으로 넘어갑니다와 연계하는 방법을 소개합니다.Databricks AI Prep Guide

실습으로 넘어갑니다

무엇을 연습해야 할지 모르시겠습니까? AI Prep 가이드(위의 AI Prep 가이드 섹션 참조)에는 "5단계 — 실습 최소 요건" 아래에 AI 챗봇이 이 시험의 목표에 맞게 매핑된 6~10개의 실습 작업으로 구성된 개인화된 체크리스트를 생성하도록 하는 프롬프트가 포함되어 있으며, 각 작업은 Databricks Free Edition에서 완료할 수 있을 만큼 충분히 작습니다.Free Edition

Databricks 문서

시험을 준비하면서 이 시험에서 다루는 플랫폼 영역에 대한 공식 Databricks 문서를 학습하십시오. 공식 Databricks 문서는 가장 최신의 권위 있는 소스입니다. 각 주제를 얼마나 깊이 학습해야 하는지 판단하고, 오래된 자료가 아닌 최신의 더 이상 사용되지 않는 기능을 학습하고 있는지 확인하는 데 활용하십시오.

공식 문서: Databricks 문서Databricks documentation

시험 개요

섹션 **1: Python** 및 **SQL**을 사용한 데이터 처리 코드 개발 **(23%)**

- 모듈식 개발 및 CI/CD 통합을 지원하기 위해 Declarative Automation Bundles(이전 명칭: Databricks Asset Bundles / DABs)에 대한 확장 가능한 Python 프로젝트 구조를 구현합니다.
- 서버리스, 파이프라인 및 번들 배포된 환경에서 외부 라이브러리(PyPI, 로컬 휠, 소스 아카이브)의 의존성 충돌 및 설치 실패에 대한 문제 해결 기법을 신청합니다.
- 주어진 제약 조건에 맞게 Pandas, Python, SQL을 사용하여 사용자 정의 함수(UDF)를 개발하며, Unity Catalog 함수를 포함합니다.
- Lakeflow Pipelines와 Auto Loader를 사용하여 스트리밍 데이터를 위한 생산 준비가 완료된 데이터 파이프라인을 생성합니다.
- UI, API, CLI를 통해 Lakeflow Jobs를 사용하여 ETL 워크로드를 생성하고 자동화합니다.
- 주어진 지연 시간, 비용 및 갱신 요구 사항에 따라 스트리밍 테이블과 구체화된 뷰 중에서 선택합니다.
- Lakeflow Pipelines에서 AUTO CDC APIs(이전 명칭: APPLY CHANGES)를 사용하여 CDC 파이프라인을 구현하며, stored_as_scd_type을 통한 SCD Type 1 및 Type 2를 포함합니다.
- 운영 제약 조건에 따라 확장 가능한 ETL을 위해 Spark Structured Streaming과 Apache Spark™ Declarative Pipelines 중에서 선택합니다.

- If/Else 조건 및 For Each 루프와 같은 제어 흐름 연산자를 사용하는 Lakeflow Jobs를 생성합니다.
- 환경 및 종속성에 적합한 compute와 구성을 선택합니다 — 서버리스 compute(서버리스 환경, 종속성 관리, 성능 모드), 고메모리 노트북 작업, 자동 최적화 설정(예: 재시도 허용 안 함) 등이 포함됩니다.
- assertDataFrameEqual, assertSchemaEqual, DataFrame.transform 및 테스트 프레임워크를 사용하여 코드 정확성을 보장하는 단위 테스트 및 통합 테스트를 개발합니다.
- 워터마크, 출력 모드, foreachBatch, 체크포인트를 포함한 Structured Streaming 상태 저장 처리 시맨틱을 신청하여 내결함성 정확히 한 번(exactly-once) 상태 복구를 구현합니다.

Section 2: 데이터 수집 및 획득 **(12%)**

- Delta Lake, Parquet, JSON, Iceberg, CSV, Binary 등 다양한 데이터 형식을 메시지 버스(Kafka, Kinesis, Pub/Sub) 및 클라우드 스토리지와 같은 다양한 소스에서 수집하는 데이터 수집 파이프라인을 개발합니다.
- Delta 또는 Iceberg를 대상 테이블 형식으로 사용하여 Lakeflow Pipelines로 증분 CDC 파이프라인을 구성합니다.
- Lakeflow Connect를 사용하여 SQL Server, MySQL, PostgreSQL을 포함한 관계형 데이터베이스 소스에서 CDC 수집 파이프라인을 구성합니다.
- 개인 정보 보호 협업을 위한 OpenSharing(D2D 및 Databricks-to-Open)과 Clean Rooms를 구현합니다.
- 지원되는 소스 시스템 전반에 걸쳐 거버넌스를 적용하여 Lakehouse Federation을 구성하고, UC 권한 및 연결 수준 자격 증명을 적용합니다.

Section 3: Data Manipulation (12%)

- Spark SQL 및 PySpark를 사용하여 대규모 데이터셋을 처리하기 위해 윈도우 함수, 조인, 집계를 포함한 고급 데이터 변환을 적용합니다.
- VARIANT 데이터 유형 및 관련 함수(예: parse_json, variant_get)를 사용하여 모델을 개발하고 반정형 데이터를 쿼리합니다.
- ai_query를 포함한 AI Functions를 신청하여 데이터 파이프라인 내에서 모델 추론을 수행하고 데이터 보강 및 분류 작업을 처리합니다.
- Lakeflow Pipelines에서 데이터 품질 기대치를 구현하여 불량 레코드를 격리, 삭제하거나 실패 처리합니다.

섹션 **4:** 모니터링 및 알림 **(10%)**

- 비용 분석, 감사 및 워크로드 모니터링을 위해 Databricks 시스템 테이블(billing, compute, access, lakeflow)을 사용합니다.
- 작업 및 파이프라인의 모니터링 및 분석을 위해 Databricks REST APIs / CLI 및 SDK를 사용합니다.
- 파이프라인 이벤트 로그를 사용하여 파이프라인 상태 및 데이터 품질을 모니터링합니다.
- Databricks Lakehouse 알림을 사용하여 관리되는 비즈니스 메트릭, 데이터 품질 및 이상 신호, 사용량 및 비용, SQL warehouse/쿼리 상태, 감사 및 보안 이벤트, AI Agent 품질, Lakeflow Job 분기를 모니터링합니다.
- Lakeflow Jobs UI와 Jobs API를 사용하여 작업 상태 및 성능 메트릭을 모니터링합니다.

섹션 **5:** 비용 및 성능 최적화**(15%)\t**

- Unity Catalog 관리형 테이블, Predictive Optimization, Liquid 클러스터링이 특정 워크로드의 운영 및 유지 관리 오버헤드를 어떻게 줄이는지 설명하십시오.
- 주어진 테이블 액세스 패턴에 적합한 Delta 최적화 기법(삭제 벡터, Liquid 클러스터링, CLUSTER BY AUTO)을 선택합니다.
- 동일한 데이터를 반복적으로 읽을 때 캐싱(Delta cache)이 쿼리 성능을 향상시키는 방식을 이해합니다.
- Change Data Feed(CDF)를 신청하여 효율적인 증분 다운스트림 처리를 위해 행 수준 변경 사항(업데이트/삭제)을 노출합니다.
- 쿼리 프로필을 사용하여 데이터 스킵 비효율성, 조인 전략, 셔플 운영 등의 성능 병목 현상을 파악합니다.
- 주어진 테이블 크기와 쿼리 패턴에 대해 Liquid 클러스터링과 파티셔닝/ZORDER를 비교합니다.

섹션 **6:** 데이터 보안 및 규정 준수 보장 **(8%)**

- Unity Catalog 보안 개체와 워크스페이스 리소스를 보호하기 위해 최소 권한 액세스 제어 목록(ACL)을 적용합니다.
- 관리되는 태그를 사용하여 속성 기반 액세스 제어(ABAC) 정책을 적용함으로써 스케일에서 행 필터 및 열 마스크를 적용합니다.
- 기밀 데이터에 익명화 및 가명화 방법(예: 해싱, 토큰화, 억제, 일반화)을 적용합니다(예: 열 마스크 및 관련 Unity Catalog 기능 사용).
- Unity Catalog 기능을 사용하여 배치 및 스트리밍 워크로드 전반에 걸쳐 PII 탐지 및 마스킹 제어를 적용하는 규정 준수 데이터 파이프라인을 구현합니다.
- Delta Lake 및 Unity Catalog 기능을 사용하여 데이터 보존 및 삭제 정책(예: GDPR 삭제권, GDPR 잊힐 권리 삭제)을 충족하는 데이터 제거 전략을 구현합니다.

섹션 **7:** 데이터 거버넌스 **(5%)**

- 보안 가능한 객체에 메타데이터를 추가하여 데이터 검색성을 향상시키는 메커니즘으로서 Unity Catalog 태그 및 주석에 대한 이해를 보여줍니다.
- 카탈로그, 스키마 및 오브젝트 전반에 걸친 액세스 제어를 관리하는 메커니즘으로서 Unity Catalog 권한 상속 모델에 대한 이해를 입증합니다.

섹션 **8:** 디버깅 및 배포 **(10%)**

- Spark UI, 클러스터 로그, 시스템 테이블, 쿼리 프로파일을 사용하여 관련 진단 정보를 파악하고 오류를 해결합니다.
- 오류를 분석하고 작업 수리 및 파라미터 재정의를 통해 실패한 작업 실행을 수정합니다.
- Declarative Automation Bundles(이전 명칭: Databricks Asset Bundles)를 사용하여 Databricks 리소스를 배포합니다.
- Databricks Git Folders(이전 명칭: Repos)를 사용하여 Git 기반 CI/CD Workflows를 구성하고 Notebooks 및 코드를 배포합니다.

섹션 **9:** 데이터 모델링 **(5%)**

- 파티셔닝을 데이터 단위에 매핑하고, 관계 액세스 패턴에 맞게 클러스터링을 정렬하며, 컴팩션을 통해 균형 잡힌 파일 크기를 유지함으로써 대규모 데이터 자산에 대한 확장 가능한 Delta/Iceberg 테이블 레이아웃을 설계합니다.
- 효율적인 쿼리 및 집계를 보장하기 위해, 사전 계산된 집계를 위한 Materialized Views와 거버넌스가 적용된 재사용 가능한 메트릭 정의를 위한 Unity Catalog Metric Views를 활용하여 분석 워크로드를 위한 차원 모델을 설계합니다.

샘플 문제

이 목적은 시험 가이드에 명시된 목표를 보여주고 각 목표에 부합하는 샘플 문제를 제공하기 위한 것입니다. 시험 가이드에는 시험에서 다룰 수 있는 목표가 나열되어 있으며, 가장 좋은 준비 방법은 위의 시험 날짜에 맞는 전체 시험 개요를 검토하는 것입니다.

아래에 있는 샘플 문제들은 새로운 시험 개요에 맞게 구성되어 있습니다**.** 시험 날짜가 **2026**년 **10**월 **9**일 이전이라면 이 문제들도 유용한 연습이 될 수 있지만**,** 일부 문제는 해당 시험 범위에 포함되지 않는 목표를 다룰 수 있습니다**.** 실제 시험에서 다루는 내용의 공식적인 목록은 시험 개요를 참고하시기 바랍니다**.**

질문 **1**

목표**:** 주어진 지연 시간**,** 비용 및 갱신 요구 사항에 따라 스트리밍 테이블과 구체화된 뷰 중 하나를 선택합니다**.**

LakeFlow 파이프라인의 골드 레이어 객체는 브론즈 소스에서 일별 주문 합계를 집계합니다. 다운스트림 BI 대시보드는 일정에 따라 이를 읽으며, 집계 결과는 새로 추가된 행뿐만 아니라 이전 날짜에 대한 늦게 도착한 업데이트를 포함한 전체 이력을 항상 반영해야 합니다.

엔지니어는 어떤 객체를 정의해야 합니까?

A. 스트리밍 테이블은 각 소스 행을 정확히 한 번 처리하며 모든 집계에 대해 가장 비용이 낮은 옵션이기 때문입니다.
B. 임시 뷰는 쿼리 시점에 항상 최신 데이터를 반영하며 스토리지 비용이 발생하지 않기 때문입니다.
C. 구체화된 뷰는 전체 집계 결과를 증분 방식으로 유지하고, 일정에 따라 새로 고칠 수 있으며, 늦게 도착한 변경 사항을 반영하기 때문입니다.
D. foreachBatch 병합을 사용하는 스트리밍 테이블입니다. Materialized Views는 집계를 수행할 수 없기 때문입니다.

질문 **2**

목표**:** 워터마크**,** 출력 모드**, foreachBatch,** 내결함성 정확히 한 번**(exactly-once)** 상태 복구를 위한 체크포인트를 포함하여 **Structured Streaming** 상태 저장 처리 의미 체계를 신청합니다**.**

Structured Streaming 쿼리는 이벤트 시간 윈도우 집계를 수행합니다. 매우 늦게 도착하는 이벤트로 인한 상태 증가를 제한해야 하며, 드라이버 장애 후 재개 시 결과가 중복 집계되지 않아야 합니다.

두 가지 요구 사항을 모두 충족하는 구성은 무엇입니까?

A. 이벤트 시간 열에 워터마크를 설정하고, 재시작 시 오프셋과 상태를 복원하기 위해 쿼리의 체크포인트에 의존합니다.
B. 정확성을 위해 모든 상태를 유지하려면 워터마크 없이 complete 출력 모드를 사용합니다.
C. spark.sql.shuffle.partitions를 늘리고 체크포인팅을 비활성화하여 복구 속도를 높입니다.
D. 집계를 foreachBatch로 감싸고 소스 오프셋을 Delta 테이블에 수동으로 커밋합니다.

질문 **3**

목표**: LakeFlow Connect**를 사용하여 **SQL Server, MySQL, PostgreSQL**을 포함한 관계형 데이터베이스 소스에서 **CDC** 수집 파이프라인을 구성합니다**.**

데이터 엔지니어는 운영 중인 PostgreSQL 데이터베이스의 삽입, 업데이트, 삭제 변경 이벤트를 최소한의 커스텀 코드로 레이크하우스에 지속적으로 복제해야 합니다.

어떤 접근 방식이 요구 사항에 적합합니까?

A. 매일 밤 JDBC 전체 테이블 읽기를 일정으로 설정하고 매일 밤 대상 테이블을 덮어씁니다.
B. PostgreSQL에 대한 LakeFlow Connect 관리형 커넥터를 구성하여 변경 데이터 캡처를 레이크하우스로 수집합니다.
C. 데이터베이스를 클라우드 스토리지에 CSV로 내보내고 Auto Loader를 사용하여 파일을 수집합니다.
D. OpenSharing을 사용하여 PostgreSQL 테이블을 레이크하우스와 직접 공유합니다.

문제 **4**

목표**: VARIANT** 데이터 유형 및 관련 함수**(**예**: parse_json, variant_get)**를 사용하여 모델을 개발하고 반정형 데이터를 쿼리합니다**.**

브론즈 테이블은 단일 문자열 열에 다양한 형태의 원시 JSON 페이로드를 저장합니다. 엔지니어는 고정 스키마를 미리 선언하지 않고도 효율적인 저장과 빠른 필드 추출을 원합니다.

어떻게 해야 합니까?

A. parse_json을 사용하여 페이로드를 VARIANT 열로 파싱하고, variant_get 및 콜론-경로 접근 방식을 사용하여 필드를 추출합니다.
B. 고정된 STRUCT 스키마를 사용하여 JSON을 키당 하나의 행으로 분해합니다.
C. 페이로드를 STRING으로 유지하고 쿼리 시 각 필드에 regexp_extract를 사용합니다.
D. 수집 전에 JSON을 엄격하고 고정된 스키마를 사용하여 Parquet으로 변환합니다.

질문 **5**

목표**:** 비용 분석**,** 감사 및 워크로드 모니터링을 위해 **Databricks** 시스템 테이블**(billing, compute, access, lakeflow)**을 사용합니다**.**

데이터 엔지니어 팀은 지난달의 DBU 지출을 특정 작업 및 SQL 웨어하우스에 귀속시키고, Unity Catalog에서 이미 사용 가능한 관리된 데이터를 활용하여 가장 비용이 많이 드는 워크로드를 파악해야 합니다.

어떤 소스를 쿼리해야 합니까?

A. 각 클러스터에 대한 Spark UI 이벤트 타임라인입니다.
B. 워크스페이스에서 다운로드한 클러스터별 드라이버 로그.
C. Unity Catalog의 compute 및 가격 시스템 테이블과 조인된 system.billing.usage 테이블입니다.
D. 어카운트 콘솔 청구 PDF 내보내기.

질문 **6**

목표**:** 주어진 테이블 액세스 패턴에 적합한 **Delta** 최적화 기법**(deletion vectors, Liquid** 클러스터링**, CLUSTER BY AUTO)**을 선택합니다**.**

데이터 엔지니어는 대규모 Delta 테이블에 개별 행을 업데이트하고 삭제하는 소규모 병합 운영이 빈번하게 발생한다는 것을 발견합니다. 팀은 변경이 있을 때마다 전체 데이터 파일을 다시 작성하지 않으면서도 읽기 속도를 빠르게 유지하고자 합니다.

쓰기 증폭(write amplification) 문제를 해결하는 기법은 무엇입니까?

A. 테이블을 기본 키로 파티션합니다.
B. deletion vector를 활성화하여 업데이트 및 삭제가 전체 파일을 즉시 다시 쓰지 않고 행 수준에서 표시되도록 합니다.
C. 모든 MERGE 후에 보존 기간을 0시간으로 설정하여 VACUUM을 실행합니다.
D. 백그라운드 파일 재작성을 방지하기 위해 Predictive Optimization을 비활성화합니다.

질문 **7**

목표**:** 거버넌스가 적용된 태그를 사용하여 속성 기반 접근 제어**(ABAC)** 정책을 신청함으로써 스케일에서 행 필터와 열 마스크를 적용합니다**.**

한 데이터 엔지니어가 PII를 포함하는 수백 개의 테이블을 보유한 조직에서 근무하고 있으며, 각 테이블을 개별적으로 수정하지 않고 현재 및 미래의 모든 테이블에서 'pii'로 태그된 열을 마스킹하는 단일 중앙 관리 규칙을 원합니다.

무엇을 구현해야 합니까?

A. 모든 기존 테이블에 대해 테이블별로 ALTER COLUMN SET MASK 문을 실행합니다.
B. 각 스키마에 행 필터 함수를 연결합니다.
C. 마스킹 로직을 사용하여 모든 기본 테이블을 복제하는 동적 뷰를 생성합니다.
D. 거버닝된 'pii' 태그가 지정된 모든 열에 열 마스크를 적용하는 ABAC 정책을 정의합니다.

질문 **8**

목표**:** 카탈로그**,** 스키마 및 객체 전반에 걸친 액세스 제어를 관리하는 메커니즘으로서 **Unity Catalog** 권한 상속 모델에 대한 이해를 증명합니다**.**

데이터 엔지니어가 analysts 그룹에 카탈로그에 대한 SELECT 권한을 부여합니다. 이후 해당 카탈로그에 새로운 스키마와 테이블이 생성됩니다.

충돌하는 권한 부여가 없다고 가정할 때, 분석가들은 새 테이블에 어떤 액세스 권한을 갖습니까?

A. 카탈로그 수준에서 부여된 권한은 나중에 생성된 것을 포함하여 그 안의 스키마와 객체에 상속되므로, 새 테이블에 대해 SELECT를 수행할 수 있습니다.
B. 각 새 테이블에 대해 SELECT 권한이 다시 부여될 때까지 액세스 권한이 없습니다.
C. 권한이 부여된 시점에 존재했던 테이블만 SELECT할 수 있습니다.
D. 새 테이블의 소유권을 자동으로 얻게 됩니다.

Question 9

목표**: Declarative Automation Bundles(**이전 명칭**: Databricks Asset Bundles)**를 사용하여 **Databricks** 리소스를 배포합니다**.**

데이터 엔지니어는 동일한 작업 및 파이프라인 정의를 소스 컨트롤에서 dev, staging, prod에 동일하게 배포하되, 환경별 재정의와 CI/CD를 적용하고자 합니다.

어떤 접근 방식이 요구 사항을 충족합니까?

A. 각 워크스페이스 UI에서 작업을 수동으로 다시 생성한 다음 JSON을 내보내고 가져옵니다.
B. databricks.yml 번들에 타겟별 재정의를 사용하여 리소스를 정의하고 `databricks bundle deploy -t <target>`으로 배포합니다.
C. 워크스페이스 import CLI만 사용하여 워크스페이스 간에 Notebooks를 복사합니다.
D. 구성 드리프트를 방지하기 위해 모든 환경에 단일 공유 워크스페이스를 사용합니다.

질문 **10**

목표**:** 효율적인 쿼리 및 집계를 보장하기 위해**,** 사전 계산된 집계를 위한 **Materialized Views**와 거버넌스가 적용된 재사용 가능한 메트릭 정의를 위한 **Unity Catalog Metric Views**를 활용하여 분석 워크로드를 위한 차원 모델을 설계합니다**.**

데이터 엔지니어링 팀이 각 대시보드에서 약간씩 다른 SQL로 "net revenue"를 계산하여 보고서 간에 일관성 없는 수치가 생성되고 있습니다. 또한 각 대시보드는 쿼리 시점에 대규모 조인과 집계를 재계산하기 때문에 속도가 느립니다. 팀은 해당 지표에 대한 하나의 신뢰할 수 있는 정의와, 대시보드가 직접 읽을 수 있는 사전 집계된 결과가 필요합니다.

엔지니어는 어떤 구현 방식을 선택해야 합니까?

A. 순수익을 계산하는 방법을 설명하는 댓글을 각 대시보드에 추가합니다.
B. 모든 분석가에게 공유 SQL 스니펫 파일에 대한 편집 액세스 권한을 부여합니다.
C. Unity Catalog 메트릭 뷰에서 순수익을 한 번 정의하여 거버넌스가 적용된 재사용 가능한 정의를 만들고, 대시보드가 읽는 사전 계산된 집계에는 Materialized Views를 사용합니다.
D. 메트릭을 각 노트북에 복사된 Python UDF로 저장합니다.

정답 키 및 해설

#	정답	해설
1	C	materialized view는 전체 집계를 증분 방식으로 유지하고 일정에 따라 새로 고칠 수 있으므로 이전 날짜에 대한 지연 도착 업데이트를 올바르게 반영합니다. 반면 streaming table은 추가된 행을 증분 방식으로 처리하며 변화하는 이력 전체에 대해 전체 집계를 다시 계산하지 않습니다.
2	A	watermark를 사용하면 엔진이 허용된 지연을 초과한 이벤트의 상태를 삭제할 수 있어(상태 증가 제한) 쿼리의 checkpoint가 오프셋과 상태를 유지하므로 재시작 시 중복 집계 없이 exactly-once로 재개됩니다.
3	B	Lakeflow Connect는 SQL Server, MySQL, PostgreSQL에서 삭제를 포함한 CDC를 최소한의 코드로 수집하는 관리형 데이터베이스 커넥터를 제공합니다. 다른 옵션은 전체 다시 로드이거나 운영 중인 RDBMS에는 적용되지 않습니다.
4	A	VARIANT 유형은 반정형 데이터를 효율적으로 저장하고 parse_json과 variant_get을 통한 스키마 유연 쿼리를 지원하므로 경직된 사전 선언 스키마를 피할 수 있습니다.
5	C	system.billing.usage(compute/pricing system tables와 조인)와 같은 system tables는 거버넌스되고 쿼리 가능한 레코드로서 비용 배분과 워크로드 분석에 이상적입니다. 다른 소스는 대규모로 쿼리할 수 없거나 거버넌스된 테이블이 아닙니다.
6	B	deletion vectors는 전체 데이터 파일을 즉시 다시 쓰지 않고 업데이트 및 삭제된 행을 표시하여, 읽기 정확성을 유지하면서 잦은 소규모 MERGE 작업으로 인한 쓰기 증폭을 줄입니다. 컴팩션이 이후에 이를 조정합니다.
7	D	거버넌스된 태그를 키로 하는 ABAC 정책은 이후에 생성된 테이블을 포함하여 'pii' 태그가 있는 모든 곳에 열 마스크를 적용하므로, 테이블별 마스크나 복제된 뷰와 달리 규칙을 중앙에서 확장할 수 있습니다.
8	A	Unity Catalog 권한은 계층 구조를 따라 하향 상속됩니다. 카탈로그에 대한 권한 부여는 해당 스키마와 그 객체에 적용되며, 권한 부여 이후에 생성된 객체도 포함됩니다.
9	B	Declarative Automation Bundle(databricks.yml)은 환경 대상과 재정의를 통해 리소스를 선언하고 CLI를 통해 배포하므로, 환경 전반에서 재현 가능하고 CI/CD에 적합한 배포를 가능하게 합니다.
10	C	metric view는 net revenue에 대한 거버넌스되고 재사용 가능한 단일 정의를 제공합니다. materialized view는 대시보드가 읽는 집계를 사전 계산하여 일관성과 성능 문제를 모두 해결합니다.