databricks

Databricks 시험 가이드

Databricks 공인 데이터 엔지니어 전문가

시험 가이드 피드백 제공

이 시험 가이드의 목적

이 시험 가이드의 목적은 시험에 대한 개요와 시험에서 다루는 내용을 제공하여 시험 준비 상태를 결정하는 데 도움이 되도록 하는 것입니다. 이 문서는 시험에 변경 사항이 있을 때마다(그리고 이러한 변경 사항이 시험에 적용되는 시기) 업데이트되어 준비할 수 있습니다. 이 버전은 **2025**년 **9**월 **30**일 현재 실시간 시험을 다룹니다**.** 시험을 치르기 **2**주 전에 다시 확인하여 최신 버전을 사용하고 있는지 확인하십시오**.**

청중 설명

Databricks Certified Data Engineering Professional 시험은 Databricks Lakehouse Platform에서 생산 등급 데이터 엔지니어링 솔루션을 구축, 최적화 및 유지 관리하는 응시자의 고급 기술을 검증합니다. 성공적인 후보자는 Delta Lake, Unity Catalog, Auto Loader, LakeFlow Spark Definitive Pipelines, Databricks Compute (serverless) LakeFlow Jobs 및 Medallion Architecture 포함과 같은 핵심 플랫폼 기능에 대한 전문성을 입증합니다. 이 인증은 안전하고 안정적이며 비용 효율적인 ETL 파이프라인을 설계하고, Python 및 SQL을 사용하여 다양한 소스의 복잡한 데이터를 처리하며, 스키마 관리, 관찰 가능성, 거버넌스 및 성능 최적화에 대한 모범 사례를 적용하는 능력을 평가합니다. 응시자는 또한 스트리밍 워크로드 구현, Workflows 오케스트레이션, DevOps 및 CI/CD 활용, Databricks CLI, REST API, 및 자산 번들과 같은 도구를 사용한 배포에 대한 테스트를 거칩니다. 이 인증을 취득한 전문가는 Databricks에서 프로덕션 준비가 완료된 데이터 엔지니어링 솔루션을 제공하는 데 필요한 지식과 실무 경험을 보유하고 있음이 입증되었으며, Lakehouse 플랫폼에 대한 1년 이상의 경험을 적극 권장합니다.

시험 정보

- 항목 수: 채점된 객관식 문제 59개
- 제한 시간: 120분
- 등록비: 미화 200달러, 현지 법률에 따라 요구되는 관련 세금
- 시험 방식: 온라인 감독
- 시험 보조 자료: 허용되지 않습니다.
- 전제 조건: 필요하지 않습니다. 강좌 출석 및 Databricks 1년의 실무 경험을 적극 권장합니다.
- 유효 기간: 2년
- 재인증: 인증 상태를 유지하려면 2년마다 재인증이 필요합니다. 재인증을 받으려면 현재 진행 중인 전체 시험에 응시해야 합니다. 시험 웹페이지의 "시험 준비" 섹션을 검토하여 시험에 다시 응시할 준비를 하십시오.
- 채점되지 않은 콘텐츠: 시험에는 나중에 사용할 수 있도록 통계 정보를 수집하기 위해 채점되지 않은 항목이 포함될 수 있습니다. 이러한 항목은 양식에서 식별되지 않으며 점수에 영향을 미치지 않으며 이 콘텐츠에 대한 추가 시간이 고려됩니다.

권장 교육

- 강사 주도 고급 Data Engineering With Databricks
- 자기 주도형 (Databricks Academy에서 사용 가능):
    - Databricks Streaming 및 LakeFlow Spark Declarative Pipelines nm
    - Databricks 데이터 개인 정보 보호
    - Databricks 성능 최적화
    - Databricks Asset Bundle을 사용한 자동화된 배포

시험 개요

섹션 **1: Python** 및 **SQL**을 사용하여 데이터 처리를 위한 코드 개발

- 개발을 위한 Python 및 도구 사용
    - Databricks Asset Bundles (DABs)에 최적화된 확장 가능한 Python 프로젝트 구조를 설계 및 구현하여 모듈식 개발, 배포 자동화 및 CI/CD 통합을 가능하게 합니다.
    - PyPI 패키지, 로컬 휠 및 소스 아카이브를 포함하여 Databricks에서 외부 타사 라이브러리 설치 및 종속성을 관리하고 문제를 해결합니다.
    - Pandas/Python UDF를 사용하여 사용자 정의 함수(UDF) 개발.
- Databricks Platform에서 Spark Declarative Pipelines, SQL 및 Apache Spark를 사용하여 ETL 파이프라인을 빌드하고 테스트합니다.
    - Spark Declarative Pipelines 및 Autoloader를 사용하여 배치 및 스트리밍 데이터에 대해 신뢰할 수 있는 프로덕션 준비 데이터 파이프라인을 구축하고 관리합니다.
    - UI/APIs/CLI를 통해 작업을 사용하여 ETL 워크로드를 생성하고 자동화합니다.
    - 구체화된 뷰와 비교하여 스트리밍 테이블의 장점과 단점을 설명합니다.
    - APPLY CHANGES APIs를 사용하여 Spark Declarative Pipelines에서 CDC를 단순화합니다.
    - Spark Structured Streaming와 Spark Declarative Pipelines를 비교하여 확장 가능한 ETL 파이프라인을 구축하기 위한 최적의 접근 방식을 결정합니다.
    - 제어 흐름 연산자 (예: if/else, foreach 등)를 사용하는 파이프라인 구성 요소를 만듭니다.
    - 환경 및 종속성에 대한 적절한 구성, notebook 작업에 대한 대용량 메모리, 재시도를 허용하지 않는 자동 최적화를 선택합니다.
    - assertDataFrameEqual, assertSchemaEqual, DataFrame.transform 및 테스트 프레임워크를 사용하여 단위 및 통합 테스트를 개발하여 기본 제공 디버거를 포함하여 코드 정확성을 보장합니다.

섹션 **2:** 데이터 수집 및 획득**:**

- 메시지 버스 및 클라우드 스토리지와 같은 다양한 소스에서 Delta Lake, Parquet, ORC, Avro, JSON, CSV, XML, 텍스트 및 바이너리를 포함한 다양한 데이터 형식을 효율적으로 수집하고 구현할 수 있도록 데이터 수집 파이프라인을 설계합니다.
- Delta를 사용하여 배치 및 스트리밍 데이터를 모두 처리할 수 있는 추가 전용 데이터 파이프라인을 만듭니다.

섹션 **3:** 데이터 변환**,** 정리 및 품질

- 효율적인 Spark SQL 및 PySpark 코드를 작성하여 창 함수, 조인 및 집계를 포함한 고급 데이터 변환을 적용하여 대규모 데이터 세트를 조작하고 분석합니다.
- 클래식 작업에서 Spark Declarative Pipelines 또는 Autoloader를 사용하여 잘못된 데이터에 대한 격리 프로세스를 개발합니다.

섹션 **4:** 데이터 공유 및 페더레이션

- Databricks를 사용하여 Databricks 배포에서 Databricks 간 공유(D2D) 또는 개방형 공유 프로토콜(D2O)을 사용하여 외부 플랫폼에 Delta Sharing 안전하게 시연합니다.
- 지원되는 소스 시스템에서 적절한 거버넌스를 사용하여 Lakehouse Federation를 구성합니다.
- Delta Sharing을 사용하여 Lakehouse의 라이브 데이터를 모든 컴퓨팅 플랫폼으로 공유하세요.

섹션 **5:** 모니터링 및 경고

- 모니터링
  - 리소스 사용률, 비용, 감사 및 워크로드 모니터링에 대한 관찰 가능성을 위해 시스템 테이블을 사용합니다.
  - 쿼리 프로파일러 UI 및 Spark UI를 사용하여 워크로드를 모니터링합니다.
  - 작업 및 파이프라인을 모니터링하기 위해 Databricks REST APIs/Databricks CLI를 사용합니다.
  - 파이프라인을 모니터링하려면 Spark Declarative Pipelines 이벤트 로그를 사용합니다.
- 경고
  - SQL 경고를 사용하여 데이터 품질을 모니터링합니다.
  - Workflows UI 및 작업 API를 사용하여 작업 상태 및 성능 문제 알림을 설정합니다.

섹션 **6:** 비용 및 성능 최적화

- Unity Catalog 관리형 테이블을 사용하여 운영 오버헤드 및 유지 관리 부담을 줄이는 방법/이유를 이해합니다.
- 삭제 벡터 및 Liquid 클러스터링와 같은 Delta 최적화 기술을 이해합니다.
- 대규모 데이터 세트에 대한 쿼리 성능을 보장하기 위해 Databricks에서 사용하는 최적화 기술 (데이터 건너뛰기, 파일 정리 등)을 이해합니다.
- Change Data Feed (CDF)를 적용하여 streaming 테이블의 특정 제한 사항을 해결하고 대기 시간을 개선합니다.
- query 프로필을 사용하여 query를 분석하고 잘못된 데이터 건너뛰기, 비효율적인 조인 유형, 데이터 섞기와 같은 병목 현상을 식별합니다.

섹션 **7:** 데이터 보안 보장 및 준수

- 데이터 보안 메커니즘 적용.
  - ACL을 사용하여 워크스페이스 개체를 보호하고 최소 권한 원칙을 적용하며, 정책 적용과 같은 원칙을 시행합니다.
  - 행 필터 및 열 마스크를 사용하여 민감한 테이블 데이터를 필터링하고 마스킹합니다.
  - 기밀 데이터에 해싱, 토큰화, 억제 및 일반화와 같은 익명화 및 가명화 방법을 적용합니다.
- 준수 보장

- 데이터 개인 정보 보호를 보장하기 위해 PII의 마스킹을 감지하고 적용하는 규정 준수 배치 및 스트리밍 파이프라인을 구현합니다.
- 데이터 보존 정책으로 규정 준수를 보장하는 데이터 제거 솔루션을 개발합니다.

섹션 **8:** 데이터 거버넌스
- 엔터프라이즈 데이터에 대한 설명/메타데이터를 만들고 추가하여 그것을 더 쉽게 검색할 수 있게 합니다.
- Unity Catalog 권한 상속 모델에 대한 이해를 보여줍니다.

섹션 **9:** 디버깅 및 배포
- 디버깅 및 문제 해결
  - Spark UI, cluster logs, 시스템 테이블 및 쿼리 프로파일을 사용하여 관련 진단 정보를 식별하여 오류를 해결합니다.
  - 오류를 분석하고 작업 복구 및 매개변수 재정의를 사용하여 실패한 작업 실행을 수정합니다.
  - Spark Declarative Pipelines 이벤트 로그 & Spark UI를 사용하여 LakeFlow를 디버깅합니다.
  - Declarative Pipelines 및 Spark 파이프라인.
- CI/CD 배포
  - Databricks Asset Bundles를 사용하여 Databricks 리소스를 빌드하고 배포합니다.
  - notebook 및 코드 배포를 위해 Databricks Git 폴더를 사용하여 Git 기반 CI/CD Workflows를 구성하고 통합합니다.

섹션 **10:** 데이터 모델링
- Delta Lake를 사용하여 확장 가능한 데이터 모델을 설계하고 구현하여 대규모 데이터 세트를 관리합니다.
- Liquid Clustering를 사용하여 데이터 레이아웃 결정을 단순화하고 쿼리 성능을 최적화합니다.
- 파티셔닝 및 ZOrder보다 Liquid Clustering를 사용할 때의 이점을 식별합니다.
- 분석 워크로드를 위한 차원 모델을 설계하여 효율적인 쿼리 및 집계를 보장합니다.

샘플 질문

이러한 문제는 이전 버전의 시험에서 사용 중지된 것입니다. 목적은 시험 가이드에 명시된 목표를 보여주고 목표에 맞는 샘플 질문을 제공하는 것입니다. 시험 가이드에는 시험에서 다룰 수 있는 목표가 나열되어 있습니다. 인증 시험을 준비하는 가장 좋은 방법은 시험 가이드의 시험 개요를 검토하는 것입니다.

질문 **1**
목표*: Delta Lake*의 카탈로그 메타스토어 작업 및 *ACID* 규정 준수 동작을 이해합니다.

아래 Delta Lake 테이블이 생성되었습니다 query:

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

원래 query에 인쇄상의 오류가 있음을 깨닫고 아래 코드가 실행되었습니다.

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

두 번째 명령을 실행한 후 어떤 결과가 발생합니까?

A. 모든 관련 파일 및 메타데이터는 단일 ACID 트랜잭션에서 삭제되고 다시 생성됩니다.
B. 테이블 이름 변경은 Delta 트랜잭션 로그에 기록됩니다.
C. 이름이 바뀐 테이블에 대해 새 Delta 트랜잭션 로그이 만들어집니다.
D. 메타스토어의 테이블 참조가 업데이트됩니다.

질문 **2**
목표*: Spark Structured Streaming* 동작을 이해하고 프로덕션 *SLA* 지원 파이프라인에 대한 최적의 접근 방식을 결정합니다.

프로덕션에 배포된 Structured Streaming 작업은 하루 중 피크 시간대에 지연이 발생했습니다. 현재 정상 실행 중에 데이터의 각 마이크로배치는 3초 이내에 처리됩니다. 하루 중 피크 시간대에는 각 마이크로배치의 실행 시간이 매우 일관되지 않아 때로는 30초를 초과하기도 합니다. 스트리밍 쓰기는 현재 10초의 트리거 간격으로 구성되어 있습니다.

다른 모든 변수를 일정하게 유지하고 레코드가 10초 이내에 처리되어야 한다고 가정하면 어떤 조정이 요구 사항을 충족합니까?

A. trigger once 옵션을 사용하고 8초마다 쿼리를 실행하도록 Databricks 작업을 구성합니다. 이렇게 하면 백로그된 모든 레코드가 각 배치로 처리됩니다.
B. 트리거 간격을 5초로 줄입니다. 배치를 더 자주 트리거하면 레코드가 백업되지 않고 대규모 배치로 인해 유출이 발생하는 것을 방지할 수 있습니다.
C. 트리거 간격을 5초로 줄입니다. 배치를 더 자주 트리거하면 유휴 실행기가 다음 배치 처리를 시작할 수 있으며 이전 배치에서 더 오래 실행되는 작업이 완료됩니다.
D. 트리거 간격은 체크포인트 디렉토리를 수정하지 않고는 수정할 수 없습니다. 현재 스트림 상태를 유지하려면 셔플 파티션 수를 늘려 병렬 처리를 최대화합니다.

질문 **3**
목표*:* 기밀 데이터에 해싱, 토큰화, 삭제, 일반화 등의 익명화 및 가명화 기법을 적용하십시오

데이터 엔지니어링 팀은 수천 개의 테이블과 뷰가 있는 엔터프라이즈 시스템을 Lakehouse로 마이그레이션하고 있습니다. 그들은 일련의 브론즈, 실버 및 골드 테이블을 사용하여 대상 아키텍처를 구현할 계획입니다. 브론즈 테이블은 프로덕션 데이터 엔지니어링 워크로드에서 거의 독점적으로 사용되는 반면, 실버 테이블은 데이터 엔지니어링 및 머신 러닝 워크로드를 모두 지원하는 데 사용됩니다. 골드 테이블은 주로 비즈니스 인텔리전스 및 보고 목적으로 사용됩니다. 개인 식별 정보 (PII)는 모든 데이터 계층에 존재하지만 실버 및 골드 수준의 모든 데이터에 대해 가명화 및 익명화 규칙이 적용됩니다.

조직은 보안 문제를 줄이는 동시에 다양한 팀 간의 협업 능력을 극대화하는 데 관심이 있습니다.

이 시스템을 구현하기 위한 모범 사례를 보여주는 진술은 무엇입니까?

A. 데이터 품질 계층에 따라 별도의 데이터베이스에서 테이블을 격리하면 데이터베이스 ACL을 통해 권한을 쉽게 관리할 수 있으며 관리되는 테이블에 대한 default 스토리지 위치를 물리적으로 분리할 수 있습니다.
B. 모든 프로덕션 테이블을 단일 데이터베이스에 저장하면 Lakehouse 전체에서 사용할 수 있는 모든 데이터 자산의 통합 뷰가 제공되므로 모든 사용자에게 이 데이터베이스에 대한 뷰 권한을 부여하여 검색 가능성이 단순화됩니다.
C. Databricks의 데이터베이스는 논리적 구성일 뿐이므로 데이터베이스 구성에 대한 선택은 Lakehouse의 보안 또는 검색 가능성에 영향을 주지 않습니다.
D. default Databricks 데이터베이스에서 작업하면 관리되는 테이블이 DBFS root에 생성되므로 관리되는 테이블로 작업할 때 가장 큰 보안이 제공됩니다.

질문 **4**
목표*: Delta Lake*를 사용하여 확장 가능한 데이터 모델을 설계하고 구현하여 대규모 데이터 세트를 관리합니다.

사용자의 콘텐츠 게시물에 대한 메타데이터를 나타내는 Delta Lake 테이블에는 다음과 같은 스키마가 있습니다.

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

위의 스키마에 따라 Delta 테이블을 분할하는 데 적합한 열은 무엇입니까?

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

질문 **5**
목표*: Unity Catalog* 권한 상속 모델에 대한 이해 입증

user_ltv라는 테이블은 다양한 팀의 데이터 분석가가 사용할 view를 만드는 데 사용됩니다. 워크스페이스의 사용자는 ACL을 사용하여 데이터 액세스를 설정하는 데 사용되는 그룹으로 구성됩니다.

user_ltv 테이블에는 다음과 같은 스키마가 있습니다.

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

다음 view 정의가 실행됩니다.

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

마케팅 그룹에 속하지 않은 분석가가 다음 query를 실행합니다:

```
SELECT * FROM email_ltv
```

이 query의 결과는 무엇일까요?

A. **email** 및 **ltv** 열만 반환됩니다. email 열에는 각 행에 문자열 " **REDACTED**" 가 포함됩니다.
B. 세 개의 열이 반환되지만 한 열의 이름 은 "**REDACTED**" 이고 null 값만 포함됩니다.
C. **email** 및 **ltv** 열만 반환되고 email 열에는 모든 null 값이 포함됩니다.
D. **email** 및 **ltv** 열은 **user_ltv**의 값과 함께 반환됩니다.

질문 6 :
목표 - 환경 및 종속성에 적합한 구성, *notebook* 작업에 대한 높은 메모리 및 재시도를 허용하지 않는 자동 최적화를 선택합니다.

비즈니스 보고 팀은 대시보드의 데이터를 매시간 업데이트하도록 요구합니다. 파이프라인에 대한 데이터를 추출, 변환 및 로드하는 파이프라인의 총 처리 시간은 10분 안에 실행됩니다.

정상적인 작동 조건을 가정할 때 가장 저렴한 비용으로 서비스 수준 계약 요구 사항을 충족하는 구성은 무엇입니까?

A. 전용 대화형 cluster에서 한 시간에 한 번 파이프라인를 실행하도록 작업을 예약합니다.
B. 새 작업 클러스터에서 한 시간에 한 번 파이프라인을 실행하도록 작업을 예약합니다.
C. 트리거 간격이 60분인 Structured Streaming 작업을 예약합니다.
D. 주어진 디렉토리에 도착하는 새 데이터가 도착할 때마다 실행되는 작업을 구성합니다.

질문 7 :
목표 - *Notebook* 개발 환경, 변수 관리 및 안전하고 구성 가능한 코드 생성을 이해합니다.

보안 팀은 외부 데이터베이스에 연결하는 데 Databricks 비밀 모듈을 활용할 수 있는지 여부를 조사하고 있습니다.

strings로 정의된 모든 Python 변수로 코드를 테스트한 후 암호 모듈에 대한 암호를 upload하고 현재 활성 사용자에 대한 올바른 권한을 구성합니다. 그런 다음 코드를 다음과 같이 수정합니다 (다른 모든 변수는 변경하지 않음).

```
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)
  )
```

위의 코드가 실행될 때 어떤 일이 발생하는지 설명하는 문은 무엇입니까?

A. 외부 테이블에 대한 연결이 성공합니다. 문자열 **"REDACTED"** 가 인쇄됩니다.
B. 외부 테이블에 대한 연결이 성공합니다. **password**의 문자열 값은 일반 텍스트로 인쇄됩니다.
C. 대화형 입력 상자가 노트북에 나타납니다; 올바른 비밀번호가 제공되면 연결이 성공하고 비밀번호가 일반 텍스트로 인쇄됩니다.
D. 대화형 입력 상자가 notebook에 나타납니다. 올바른 비밀번호가 제공되면 연결이 성공하고 인코딩된 비밀번호가 DBFS에 저장됩니다.

질문 8:
목표:대규모 데이터 세트에 대한 쿼리 성능을 보장하기 위해 *Databricks*에서 사용하는 최적화 기술*(*데이터 건너뛰기, 파일 정리 등*)*을 이해합니다.

데이터 수집 작업에는 대상 파트 파일 크기가 512MB인 Parquet에 1TB JSON 데이터세트를 기록해야 합니다. Delta Lake 대신 Parquet가 사용되기 때문에 자동 최적화 및 자동 압축과 같은 기본 제공 파일 크기 조정 기능을 사용할 수 없습니다.

데이터를 섞지 않고 최상의 성능을 얻을 수 있는 전략은 무엇입니까?

A. 데이터를 수집하고, 좁은 변환을 실행하고, 2,048개의 파티션 (1TB*1024*1024/512)으로 다시 분할한 다음, Parquet에 씁니다.
B. **spark.sql.adaptive.advisoryPartitionSizeInBytes**를 512MB 바이트로 설정 하고, 데이터를 수집하고, 좁은 변환을 실행하고, 2,048개의 파티션(1TB*1024*1024/512)으로 병합한 다음, Parquet에 씁니다.
C. **spark.sql.files.maxPartitionBytes**를 512MB로 설정 하고, 데이터를 수집하고, 좁은 변환을 실행한 다음, Parquet에 씁니다.
D. **spark.sql.shuffle.partitions**를 2,048개의 파티션(1TB*1024*1024/512)으로 설정 하고, 데이터를 수집하고, 좁은 변환을 실행하고, 데이터를(자동으로 다시 분할함) 정렬하여 데이터를 최적화한 다음, Parquet에 씁니다.

질문 9 :
목표*: Delta Lake* 복제를 적용하여 얕은 복제와 심층 복제가 소스/대상 테이블과 상호 작용하는 방법을 알아봅니다.

마케팅 팀은 집계 테이블의 데이터를 영업 조직과 공유하려고 하지만 팀에서 사용하는 필드 이름이 일치하지 않으며 여러 마케팅 관련 필드가 영업 조직에 대해 승인되지 않았습니다.

단순성을 강조하면서 상황을 해결하는 솔루션은 무엇입니까?

A. 영업 팀에 대해 승인된 필드만 선택하여 마케팅 테이블에 뷰를 만듭니다. 별칭을 판매 명명 규칙에 따라 표준화해야 하는 모든 필드의 이름을 지정합니다.
B. 필요한 스키마로 새 테이블을 만들고 Delta Lake의 DEEP CLONE 기능을 사용하여 한

테이블에 커밋된 변경 사항을 해당 테이블에 동기화합니다.

C. CTAS 문을 사용하여 마케팅 테이블에서 파생 테이블을 만듭니다. 변경 사항을 전파하도록 생산 작업을 구성합니다.
D. 현재 생산 파이프라인에 병렬 테이블 쓰기를 추가하여 필요에 따라 마케팅 테이블에서 다른 새 판매 테이블을 업데이트합니다.

질문 10:
목표*:* 여러 종속성이 있는 다중 태스크 작업 생성

Databricks 작업은 세 가지 태스크로 구성되었으며 각 작업은 Databricks notebook. 태스크 A는 다른 태스크에 의존하지 않습니다. 태스크 B와 C는 병렬로 실행되며 각각 태스크 A에 대한 직렬 종속성이 있습니다.

태스크 A와 B가 성공적으로 완료되었지만 예약된 실행 중에 태스크 C가 실패하는 경우 결과 상태는 어떻게 됩니까?

A. 태스크 A 및 B와 관련된 노트북에 표현된 모든 논리가 성공적으로 완료되었습니다. 태스크 C의 일부 작업이 성공적으로 완료되었을 수 있습니다.
B. 모든 태스크가 성공적으로 완료되지 않으면 변경 사항이 Lakehouse에 커밋되지 않습니다. 태스크 C가 실패했기 때문에 모든 커밋이 자동으로 롤백됩니다.
C. 태스크 A 및 B와 관련된 노트북에 표현된 모든 논리가 성공적으로 완료되었습니다. 태스크 C의 모든 변경 사항은 태스크 실패로 인해 롤백됩니다.
D. 모든 태스크는 종속성 그래프로 관리되므로 모든 태스크가 성공적으로 완료될 때까지 변경 사항이 Lakehouse에 커밋되지 않습니다.

답변
질문 1: D
질문 2: B
질문 3: A
질문 4: C
질문 5: A
질문 6: B
질문 7: A
질문 8: C
질문 9: A
질문 10:A