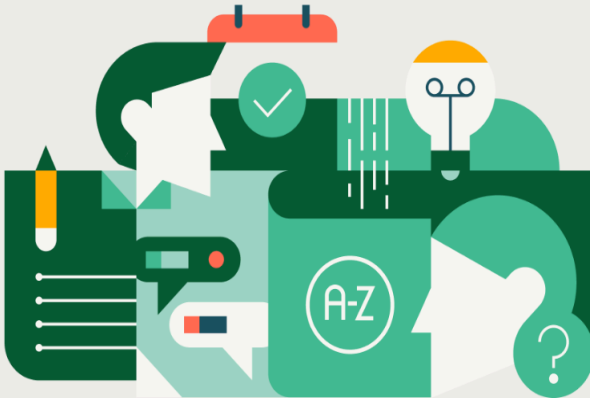
databricks

Databricks 시험 가이드

Databricks Certified Data Engineer Professional



시험 가이드 피드백 제공

이 시험 가이드의 목적

이 시험 가이드는 Databricks Certified Data Engineer Professional 시험에 대한 개요와 시험 범위를 제공하여 시험 준비 상태를 판단하는 데 도움을 줍니다. 이 문서는 시험에 변경 사항이 있을 때마다(그리고 해당 변경 사항이 시험에 적용되는 시점에) 업데이트되므로 미리 대비할 수 있습니다. 시험 **2**주 전에 다시 확인하여 최신 버전을 확보하시기 바랍니다**.** 이 버전은 **2025**년 **11**월 **30**일 기준으로 현재 운영 중인 버전을 다룹니다**.**

대상자 설명

Databricks Certified Data Engineering Professional 시험은 Databricks Data Intelligence Platform에서 프로덕션 수준의 데이터 엔지니어링 솔루션을 구축, 최적화 및 유지 관리하는 응시자의 고급 역량을 검증합니다. 합격자는 Delta Lake, Unity Catalog, Auto Loader, Lakeflow Spark Declarative Pipelines, Databricks Compute(서버리스 포함), Lakeflow Jobs, Medallion Architecture와 같은 핵심 플랫폼 기능 전반에 걸친 전문성을 입증합니다. 이 자격증은 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, 워크플로 오케스트레이션, DevOps 및 CI/CD 활용, Databricks CLI, REST API, Asset Bundles와 같은 도구를 사용한 배포 능력에 대해서도 평가받습니다. 이 자격증 시험에 합격한 사람은 Databricks 및 관련 도구를 사용하여 고급 데이터 엔지니어링 작업을 수행할 수 있을 것으로 기대됩니다.

시험 정보

- 문항 수: 채점 대상 객관식 문제 59개
- 제한 시간: 120분
- 응시료: USD 200 및 현지 법률에 따라 부과되는 세금
- 응시 방식: 온라인 감독 및 시험 센터 감독
- 시험 보조 자료: API 문서를 포함하여 어떠한 시험 보조 자료도 제공되지 않습니다.
- 사전 요건: 필수 요건은 없으나, 관련 과정 수강 및 시험 가이드에 명시된 데이터 엔지니어링 작업에 대한 1년간의 실무 경험을 적극 권장합니다.
- 유효 기간: 2년

- 재인증: 인증 상태를 유지하려면 2년마다 재인증이 필요합니다. 재인증을 받으려면 현재 버전의 시험에 응시해야 합니다. 재인증 시험을 준비하려면 아래의 "시험 준비하기" 섹션을 검토하시기 바랍니다.
- 채점 제외 콘텐츠: 시험에는 향후 사용을 위한 통계 정보를 수집하기 위해 채점되지 않는 문항이 포함될 수 있습니다. 이러한 문항은 시험지에 표시되지 않으며 점수에 영향을 미치지 않고, 이 콘텐츠를 위한 추가 시간이 반영됩니다.

권장 교육

- 강사 주도형: Advanced Data Engineering With Databricks
- 자기 주도형(Databricks Academy에서 제공):
    - Databricks Streaming and Lakeflow Spark Declarative Pipelines
    - Databricks Data Privacy
    - Databricks Performance Optimization
    - Automated Deployment with Databricks Asset Bundle

시험 개요

섹션 **1: Python** 및 **SQL**을 사용한 데이터 처리 코드 개발

- 개발을 위한 Python 및 도구 사용
- 모듈식 개발, 배포 자동화 및 CI/CD 통합을 가능하게 하는, Databricks Asset Bundles(DABs)에 최적화된 확장 가능한 Python 프로젝트 구조를 설계하고 구현합니다.
- PyPI 패키지, 로컬 wheel 및 소스 아카이브를 포함하여 Databricks의 외부 서드파티 라이브러리 설치 및 종속성을 관리하고 문제를 해결합니다.
- Pandas/Python UDF를 사용하여 사용자 정의 함수(UDF)를 개발합니다.
- Databricks Platform에서 Lakeflow Spark Declarative Pipelines, SQL 및 Apache Spark를 사용하여 ETL 파이프라인 구축 및 테스트
- Lakeflow Spark Declarative Pipelines 및 Autoloader를 사용하여 배치 및 스트리밍 데이터를 위한 안정적이고 프로덕션에 즉시 사용할 수 있는 데이터 파이프라인을 구축하고 관리합니다.
- UI/API/CLI를 통해 Jobs를 사용하여 ETL 워크로드를 생성하고 자동화합니다.
- 머티리얼라이즈드 뷰와 비교하여 스트리밍 테이블의 장점과 단점을 설명합니다.
- APPLY CHANGES API를 사용하여 Lakeflow Spark Declarative Pipelines에서 CDC를 단순화합니다.
- Spark Structured Streaming과 Lakeflow Spark Declarative Pipelines를 비교하여 확장 가능한 ETL 파이프라인을 구축하기 위한 최적의 접근 방식을 결정합니다.
- 제어 흐름 연산자(예: if/else, for/each 등)를 사용하는 파이프라인 구성 요소를 생성합니다.
- 환경 및 종속성, 노트북 작업을 위한 고용량 메모리, 재시도를 허용하지 않는 자동 최적화에 적합한 구성을 선택합니다.
- 다음을 사용하여 단위 및 통합 테스트를 개발합니다: assertDataFrameEqual,
- assertSchemaEqual, DataFrame.transform 및 테스트 프레임워크를 사용하여 내장 디버거를 포함해 코드의 정확성을 보장합니다.

섹션 **2:** 데이터 수집 및 취득**:**

- 다양한 데이터를 효율적으로 수집하기 위한 데이터 수집 파이프라인을 설계하고 구현합니다
- Delta Lake, Parquet, ORC, AVRO, JSON, CSV, XML, Text 및 Binary를 포함한 형식으로,
- 메시지 버스 및 클라우드 스토리지와 같은 다양한 소스로부터 수집합니다.
- Delta를 사용하여 배치 및 스트리밍 데이터를 모두 처리할 수 있는 추가 전용(append-only) 데이터 파이프라인을 생성합니다.

섹션 **3:** 데이터 변환**,** 정제 및 품질

- 윈도우 함수, 조인 및 집계를 포함한 고급 데이터 변환을 적용하는 효율적인 Spark SQL 및 PySpark 코드를 작성하여 대규모 데이터셋을 조작하고 분석합니다.
- Lakeflow Spark Declarative Pipelines 또는 클래식 작업의 autoloader를 사용하여 잘못된 데이터를 격리하는 프로세스를 개발합니다.

섹션 **4:** 데이터 공유 및 페더레이션

- Databricks to Databricks Sharing(D2D)을 사용하여 Databricks 배포 간에, 또는 개방형 공유 프로토콜(D2O)을 사용하여 외부 플랫폼으로 delta sharing을 안전하게 수행하는 방법을 시연합니다.
- 지원되는 소스 시스템 전반에 걸쳐 적절한 거버넌스와 함께 Lakehouse Federation을 구성합니다.
- Delta Share를 사용하여 Lakehouse의 실시간 데이터를 모든 컴퓨팅 플랫폼으로 공유합니다.

섹션 **5:** 모니터링 및 경고

- 모니터링
    - 시스템 테이블을 사용하여 리소스 사용률, 비용, 감사 및 워크로드 모니터링에 대한 관찰 가능성을 확보합니다.
    - Query Profiler UI 및 Spark UI를 사용하여 워크로드를 모니터링합니다.
    - Databricks REST API/Databricks CLI를 사용하여 작업 및 파이프라인을 모니터링합니다.
    - Lakeflow Spark Declarative Pipelines Event Logs를 사용하여 파이프라인을 모니터링합니다.
- 경고
    - SQL Alerts를 사용하여 데이터 품질을 모니터링합니다.
    - Lakeflow Jobs UI 및 Jobs API를 사용하여 작업 상태 및 성능 문제에 대한 알림을 설정합니다.

섹션 **6:** 비용 및 성능 최적화

- Unity Catalog 관리형 테이블을 사용하면 운영
- 오버헤드 및 유지 관리 부담이 어떻게/왜 줄어드는지 이해합니다.
- deletion vector 및 liquid clustering과 같은 delta 최적화 기법을 이해합니다.
- 대규모 데이터셋에 대한 쿼리 성능을 보장하기 위해 Databricks가 사용하는 최적화

기법(데이터 건너뛰기, 파일 프루닝 등)을 이해합니다.

- Change Data Feed(CDF)를 적용하여 스트리밍 테이블의 특정 제한 사항을 해결하고 지연 시간을 개선합니다.
- 쿼리 프로파일을 사용하여 쿼리를 분석하고 비효율적인 데이터 건너뛰기, 비효율적인 조인 유형, 데이터 셔플링과 같은 병목 지점을 식별합니다.

섹션 **7:** 데이터 보안 및 규정 준수 보장

- 데이터 보안 메커니즘 적용.
    - ACL을 사용하여 Workspace 객체를 보호하고, 최소 권한 원칙을 적용하며, 최소 권한 및 정책 적용과 같은 원칙을 시행합니다.
    - 행 필터 및 열 마스크를 사용하여 민감한 테이블 데이터를 필터링하고 마스킹합니다. ○ 해싱, 토큰화, 억제(Suppression) 및 일반화(generalisation)와 같은 익명화 및 가명화 방법을 기밀 데이터에 적용합니다.
- 규정 준수 보장
    - PII를 감지하고 마스킹을 적용하여 데이터 프라이버시를 보장하는, 규정을 준수하는 배치 및 스트리밍 파이프라인을 구현합니다.
    - 데이터 보존 정책 준수를 보장하는 데이터 삭제(purging) 솔루션을 개발합니다.

섹션 **8:** 데이터 거버넌스

- 엔터프라이즈 데이터에 대한 설명/메타데이터를 생성하고 추가하여 검색 가능성을 높입니다.
- Unity Catalog 권한 상속 모델에 대한 이해를 입증합니다.

섹션 **9:** 디버깅 및 배포

- 디버깅 및 문제 해결
    - Spark UI, 클러스터 로그, 시스템 테이블 및 쿼리 프로파일을 사용하여 오류를 해결하기 위한 관련 진단 정보를 식별합니다.
    - 오류를 분석하고 작업 복구(job repair) 및 매개변수 재정의를 통해 실패한 작업 실행을 개선합니다.
    - Lakeflow Spark Declarative Pipelines 이벤트 로그와 Spark UI를 사용하여 Lakeflow Spark Declarative Pipelines 및 Spark 파이프라인을 디버깅합니다.
- ● CI/CD 배포
    - Databricks Asset Bundles를 사용하여 Databricks 리소스를 구축하고 배포합니다.
    - 노트북 및 코드 배포를 위해 Databricks Git Folders를 사용하여 Git 기반 CI/CD 워크플로를 구성하고 통합합니다.

섹션 **10:** 데이터 모델링

- Delta Lake를 사용하여 대규모 데이터셋을 관리하기 위한 확장 가능한 데이터 모델을 설계하고 구현합니다.
- Liquid Clustering을 사용하여 데이터 레이아웃 결정을 단순화하고 쿼리 성능을 최적화합니다.
- Partitioning 및 ZOrder보다 liquid Clustering을 사용하는 이점을 식별합니다.
- 효율적인 쿼리 및 집계를 보장하는 분석 워크로드용 차원 모델(Dimensional Model)을

설계합니다.

샘플 문제

이 문제들은 이전 버전의 시험에서 더 이상 사용되지 않는 문제입니다. 그 목적은 시험 가이드에 명시된 목표를 보여주고, 해당 목표에 부합하는 샘플 문제를 제공하는 데 있습니다. 시험 가이드에는 시험에서 다룰 수 있는 목표가 나열되어 있습니다. 자격증 시험을 준비하는 가장 좋은 방법은 시험 가이드의 시험 개요를 검토하는 것입니다.

문제 **1**
목표*: Delta Lake*의 *catalog-metastore* 작업 및 *ACID* 준수 동작을 이해합니다.

다음 쿼리로 Delta Lake 테이블이 생성되었습니다:

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

원본 쿼리에 오타가 있음을 인지하여 아래 코드가 실행되었습니다: ALTER TABLE prod.sales_by_stor RENAME TO prod.sales_by_store 두 번째 명령을 실행한 후 어떤 결과가 발생합니까?

A. 관련된 모든 파일과 메타데이터가 삭제되고 단일 ACID 트랜잭션으로 다시 생성됩니다.
B. 테이블 이름 변경이 Delta 트랜잭션 로그에 기록됩니다.
C. 이름이 변경된 테이블에 대해 새로운 Delta 트랜잭션 로그가 생성됩니다..
D. metastore의 테이블 참조가 업데이트됩니다.

문제 **2**
목표*: Spark Structured Streaming* 동작을 이해하고 프로덕션 *SLA*를 충족하는 파이프라인을 위한 최적의 접근 방식을 결정합니다.

프로덕션에 배포된 Structured Streaming 작업이 하루 중 피크 시간대에 지연을 겪고 있습니다. 현재 정상 실행 중에는 각 데이터 마이크로배치가 3초 이내에 처리됩니다. 하루 중 피크 시간대에는 각 마이크로배치의 실행 시간이
매우 불규칙해져 때때로 30초를 초과합니다. 스트리밍 쓰기는 현재 10초 트리거 간격으로 구성되어 있습니다.

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

A. trigger once 옵션을 사용하고 8초마다 쿼리를 실행하도록 Databricks 작업을 구성합니다. 이렇게 하면 밀린 모든 레코드가 각 배치에서 처리됩니다. B. 트리거 간격을 5초로 줄입니다. 배치를 더 자주 트리거하면 레코드가 밀리는 것과 대용량 배치로 인한 스필(spill)을 방지할 수 있습니다.

C. 트리거 간격을 5초로 줄입니다. 배치를 더 자주 트리거하면 이전 배치의 장기 실행 작업이 완료되는 동안 유휴 실행자(executor)가 다음 배치 처리를 시작할 수 있습니다.
D. 체크포인트 디렉터리를 수정하지 않고는 트리거 간격을 변경할 수 없습니다. 현재 스트림 상태를 유지하려면 셔플 파티션 수를 늘려 병렬성을 최대화합니다.

문제 **3**
목표: *Delta Lake*를 사용하여 대규모 데이터셋을 관리하기 위한 확장 가능한 데이터 모델을 설계하고 구현합니다.

사용자의 콘텐츠 게시물에 대한 메타데이터를 나타내는 Delta Lake 테이블의 스키마는 다음과 같습니다:

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

위 스키마를 기준으로 할 때, Delta 테이블을 파티셔닝하기에 적합한 후보 열은 무엇입니까?

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

문제 **4**
목표: *Unity Catalog* 권한 상속 모델에 대한 이해를 입증합니다

user_ltv라는 이름의 테이블이 여러 팀의 데이터 분석가가 사용할 뷰를 생성하는 데 사용되고 있습니다. workspace의 사용자는 그룹으로 구성되며, 이 그룹은 ACL을 사용한 데이터 액세스 설정에 사용됩니다.
user_ltv 테이블의 스키마는 다음과 같습니다:
`email STRING, age INT, ltv INT`
다음 뷰 정의가 실행됩니다:

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

marketing 그룹의 구성원이 아닌 분석가가 다음 쿼리를 실행합니다: SELECT * FROM email_ltv

이 쿼리의 결과는 무엇입니까?

A. email 및 ltv 열만 반환되며, email 열에는 각 행에 문자열 "REDACTED"가 포함됩니다.
B. 세 개의 열이 반환되지만, 한 열의 이름은 "REDACTED"이며 null 값만 포함합니다.
C. email 및 ltv 열만 반환되며, email 열에는 모두 null 값이 포함됩니다.
D. email 및 ltv 열이 user_ltv의 값과 함께 반환됩니다.

문제 **5**
목표- 환경 및 종속성, 노트북 작업을 위한 고용량 메모리, 재시도를 허용하지 않는 자동 최적화에 적합한 구성을 선택합니다.

비즈니스 보고 팀은 대시보드용 데이터가 매시간 업데이트되기를 요구합니다. 실행 시 데이터를 추출, 변환 및 로드하는 이 파이프라인의 총 처리 시간은 10분입니다.
정상적인 운영 조건을 가정할 때, 어떤 구성이 가장 낮은 비용으로 서비스 수준 계약(SLA) 요구 사항을 충족합니까?

A. 전용 대화형 클러스터에서 한 시간에 한 번 파이프라인을 실행하도록 작업을 예약합니다.
B. 새 작업 클러스터에서 한 시간에 한 번 파이프라인을 실행하도록 작업을 예약합니다.
C. 60분 트리거 간격으로 Structured Streaming 작업을 예약합니다.
D. 지정된 디렉터리에 새 데이터가 도착할 때마다 실행되는 작업을 구성합니다.

문제 **6**
목표- 노트북 개발 환경, 변수 관리 및 안전하고 구성 가능한 코드 작성을 이해합니다.

보안 팀은 외부 데이터베이스에 연결하는 데 Databricks secrets 모듈을 활용할 수 있는지 검토하고 있습니다.
모든 Python 변수를 문자열로 정의하여 코드를 테스트한 후, 비밀번호를 secrets 모듈에 업로드하고 현재 활성 사용자에 대해 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)
)
```

이 코드를 실행하면 어떻게 됩니까?

A. 외부 테이블에 대한 연결이 성공하고, 문자열 "REDACTED"가 출력됩니다.
B. 외부 테이블에 대한 연결이 성공하고, 비밀번호의 문자열 값이 일반 텍스트로 출력됩니다.
C.. 노트북에 대화형 입력 상자가 나타나고, 올바른 비밀번호가 제공되면 연결이 성공하며 비밀번호가 일반 텍스트로 출력됩니다.
D. 노트북에 대화형 입력 상자가 나타나고, 올바른 비밀번호가 제공되면 연결이 성공하며 인코딩된 비밀번호가 DBFS에 저장됩니다.

문제 **7**
목표: 대규모 데이터셋에 대한 쿼리 성능을 보장하기 위해 *Databricks*가 사용하는 최적화 기법*(*데이터 건너뛰기, 파일 프루닝 등*)*을 이해합니다

데이터 수집 작업에서 1TB JSON 데이터셋을 목표 파트 파일 크기 512MB로 Parquet에 기록해야 합니다. Delta Lake 대신 Parquet를 사용하므로 Auto-Optimize 및 Auto-Compaction과 같은 내장 파일 크기 조정 기능을 사용할 수 없습니다.

데이터를 재배치하지 않고 작동하는 접근 방식은 무엇입니까?

A. 데이터를 수집하고, narrow 변환을 실행한 다음, 2,048개 파티션(1TB*1024*1024/512)으로 repartition한 후 Parquet에 기록합니다.
B. spark.sql.adaptive.advisoryPartitionSizeInBytes를 512MB로 설정하고, 데이터를 수집하고, narrow 변환을 실행한 다음, 2,048개 파티션으로 coalesce합니다
(1TB*1024*1024/512) 그런 다음 Parquet에 기록합니다.
C. spark.sql.files.maxPartitionBytes를 512MB로 설정하고, 데이터를 수집하고, narrow 변환을 실행한 다음 Parquet에 기록합니다.
D. spark.sql.shuffle.partitions를 2,048개 파티션(1TB*1024*1024/512)으로 설정하고, 데이터를 수집하고, narrow 변환을 실행하고, 데이터를 정렬하여 최적화한(이는 데이터를 자동으로 repartition함) 다음 Parquet에 기록합니다.

문제 **8**
목표*: Delta Lake clone*을 적용하여 *shallow clone*과 *deep clone*이 소스*/*대상 테이블과 어떻게 상호 작용하는지 학습합니다.

marketing 팀은 집계 테이블의 데이터를 sales 조직과 공유하고자 하지만, 두 팀이 사용하는 필드 이름이 일치하지 않으며 다수의 marketing 전용 필드는 sales 조직에 대해 승인되지 않았습니다. 단순성을 강조하면서 이 상황을 해결하는 솔루션은 무엇입니까?

A. sales 팀에 승인된 필드만 선택하는 뷰를 marketing 테이블에 생성하고, sales 명명 규칙으로 표준화해야 하는 필드의 이름에 별칭(alias)을 지정합니다.
B. 필요한 스키마로 새 테이블을 생성하고 Delta Lake의 DEEP CLONE 기능을 사용하여 한 테이블에 커밋된 변경 사항을 해당 테이블에 동기화합니다. C. CTAS 문을 사용하여 marketing 테이블에서 파생 테이블을 생성한 다음, 변경 사항을 전파하도록 프로덕션 작업을 구성합니다.

D. 현재 프로덕션 파이프라인에 병렬 테이블 쓰기를 추가하여, marketing 테이블과 필요에 따라 달라지는 새 sales 테이블을 업데이트합니다.

문제 **9**
목표: 여러 종속성을 가진 멀티 태스크 작업을 생성합니다.

세 개의 태스크로 구성된 Databricks 작업이 있으며, 각 태스크는 Databricks 노트북입니다. 태스크 A는 다른 태스크에 의존하지 않습니다. 태스크 B와 C는 병렬로 실행되며, 각각 태스크 A에 대한 직렬 종속성을 가집니다.

예약된 실행 중에 태스크 A와 B는 성공적으로 완료되지만 태스크 C가 실패할 경우 결과 상태는 무엇입니까?

A. 태스크 A와 B에 연결된 노트북에 표현된 모든 로직이 성공적으로 완료되며, 태스크 C의 일부 작업은 성공적으로 완료되었을 수 있습니다.
B. 모든 태스크가 성공적으로 완료되지 않는 한 어떠한 변경 사항도 Lakehouse에 커밋되지 않으며, 태스크 C가 실패했으므로 모든 커밋이 자동으로 롤백됩니다. C. 태스크 A와 B에 연결된 노트북에 표현된 모든 로직이 성공적으로 완료되며, 태스크 C에서 이루어진 모든 변경 사항은 태스크 실패로 인해 롤백됩니다.
D. 모든 태스크가 종속성 그래프로 관리되므로, 모든 태스크가 성공적으로 완료될 때까지 어떠한 변경 사항도 Lakehouse에 커밋되지 않습니다.

Answers

문제 *1: D*
문제 *2: B*
문제 *3: C*
문제 *4: A*
문제 *5: B*
문제 *6: A*
문제 *7: C*
문제 *8: A*
문제 *9 :A*