Up-to-date Practice Test with Latest Questions and Answers covering latest syllabus and topics of the exam. Makes you ready to face actual exam.

DCDEP Practice Questions
DCDEP Practice Test
DCDEP Practice Exam
DCDEP Exam Questions
DCDEP Study Guide

killexams.com

Databricks

DCDEP

Databricks Certified Data Engineer Professional

ORDER FULL VERSION

https://killexams.com/pass4sure/exam-detail/DCDEP

Question: 524

Objective: Assess the impact of a MERGE operation on a Delta Lake table

A Delta Lake table prod.customers has columns customer_id, name, and last_updated. A data engineer runs the following MERGE operation to update records from a source DataFrame updates_df:

```
MERGE INTO prod.customers AS target
USING updates_df AS source
ON target.customer_id = source.customer_id
WHEN MATCHED THEN UPDATE SET
name = source.name,
last_updated = source.last_updated
WHEN NOT MATCHED THEN INSERT
(customer_id, name, last_updated)
VALUES
(source.customer_id, source.name, source.last_updated)
```

If updates_df contains a row with customer_id = 100, but prod.customers has multiple rows with customer_id = 100, what happens?

A. The operation succeeds, updating all matching rows with the same values.
B. The operation fails due to duplicate customer_id values in the target.
C. The operation skips the row with customer_id = 100.
D. The operation inserts a new row for customer_id = 100.
E. The operation updates only the first matching row.

Answer: B

Explanation: Delta Lake's MERGE operation requires that the ON condition matches at most one row in the target table for each source row. If multiple rows in prod.customers match customer_id = 100, the operation fails with an error indicating ambiguous matches.

Question: 525

A data engineer enables Change Data Feed (CDF) on a Delta table orders to propagate

changes to a target table orders_sync. The CDF is enabled at version 12, and the pipeline processes updates and deletes. Which query correctly applies CDC changes?

A. spark.readStream.option("readChangeFeed", "true").option("startingVersion", 12).table("orders").writeStream.outputMode("append").table("orders_sync")
B. spark.readStream.option("readChangeFeed", "true").option("startingVersion", 12).table("orders").groupBy("order_id").agg(max("amount")).writeStream.outputMode("append").t
C. spark.read.option("readChangeFeed", "true").table("orders").writeStream.outputMode("update").table("orders_sync")
D. spark.readStream.option("readChangeFeed", "true").table("orders").writeStream.outputMode("complete").table("orders_sync")
E. spark.readStream.option("readChangeFeed", "true").option("startingVersion", 12).table("orders").writeStream.foreachBatch(lambda batch, id: spark.sql("MERGE INTO orders_sync USING batch ON orders_sync.order_id = batch.order_id WHEN MATCHED AND batch._change_type = 'update' THEN UPDATE SET * WHEN MATCHED AND batch._change_type = 'delete' THEN DELETE WHEN NOT MATCHED AND batch._change_type = 'insert' THEN INSERT *"))

Answer: E

Explanation: CDF processing uses spark.readStream.option("readChangeFeed", "true") with startingVersion set to 12. The foreachBatch method with a MERGE statement applies inserts, updates, and deletes based on _change_type.

Question: 526

A data engineer is analyzing a Spark job that processes a 1TB Delta table using a cluster with 8 worker nodes, each with 16 cores and 64GB memory. The job involves a complex join operation followed by an aggregation. In the Spark UI, the SQL/DataFrame tab shows a query plan with a SortMergeJoin operation taking 80% of the total execution time. The Stages tab indicates one stage has 200 tasks, but 10 tasks are taking significantly longer, with high GC Time and Shuffle Write metrics. Which optimization should the engineer prioritize to reduce execution time?

A. Increase the number of worker nodes to 16 to distribute tasks more evenly
B. Set spark.sql.shuffle.partitions to 400 to increase parallelism
C. Enable Adaptive Query Execution (AQE) with spark.sql.adaptive.enabled=true
D. Increase spark.executor.memory to 128GB to reduce garbage collection
E. Use OPTIMIZE and ZORDER on the Delta table to improve data skipping

Answer: C

Explanation: The high execution time of the SortMergeJoin and skewed tasks with high GC Time and Shuffle Write suggest data skew and shuffle bottlenecks. Enabling Adaptive Query Execution (AQE) with spark.sql.adaptive.enabled=true allows Spark to dynamically adjust the number of partitions, optimize join strategies, and handle skew by coalescing small partitions or splitting large ones. This is more effective than increasing nodes (which increases costs without addressing skew), changing shuffle partitions manually (which may not address skew dynamically), increasing memory (which may not solve shuffle issues), or using OPTIMIZE and ZORDER (which improves data skipping but not join performance directly).

Question: 527

A data engineer is deduplicating a streaming DataFrame orders with columns order_id, customer_id, and event_time. Duplicates occur within a 10-minute window. The deduplicated stream should be written to a Delta table orders_deduped in append mode. Which code is correct?

A. orders.dropDuplicates("order_id").withWatermark("event_time", "10 minutes").writeStream.outputMode("append").table("orders_deduped")
B. orders.dropDuplicates("order_id", "event_time").writeStream.outputMode("append").table("orders_deduped")
C. orders.withWatermark("event_time", "10 minutes").groupBy("order_id").agg(max("event_time")).writeStream.outputMode("complete").table
D. orders.withWatermark("event_time", "10 minutes").dropDuplicates("order_id").writeStream.outputMode("append").table("orders_deduped")
E. orders.withWatermark("event_time", "10 minutes").distinct().writeStream.outputMode("update").table("orders_deduped")

Answer: D

Explanation: Deduplication requires withWatermark("event_time", "10 minutes") followed by dropDuplicates("order_id") to remove duplicates within the 10-minute window. The append mode writes deduplicated records to the Delta table.

Question: 528

A data engineer is optimizing a Delta table logs with 1 billion rows, partitioned by log_date. Queries filter by log_type and user_id. The engineer runs OPTIMIZE logs ZORDER BY (log_type, user_id) but notices minimal performance improvement. What is the most likely cause?

A. The table is too large for Z-ordering
B. The table is not vacuumed
C. log_type and user_id have low cardinality
D. Z-ordering is not supported on partitioned tables

Answer: C

Explanation: Z-ordering is less effective for low-cardinality columns like log_type and user_id, as it cannot efficiently co-locate data. Table size doesn't prevent Z-ordering. Vacuuming removes old files but doesn't affect Z-ordering. Z-ordering is supported on partitioned tables.

Question: 529

A Delta Lake table logs_data with columns log_id, device_id, timestamp, and event is partitioned by timestamp (year-month). Queries filter on event and timestamp ranges. Performance is poor due to small files. Which command optimizes the table?

A. OPTIMIZE logs_data ZORDER BY (event, timestamp)
B. ALTER TABLE logs_data SET TBLPROPERTIES ('delta.targetFileSize' = '512MB')
C. REPARTITION logs_data BY (event)
D. OPTIMIZE logs_data PARTITION BY (event, timestamp)
E. VACUUM logs_data RETAIN 168 HOURS

Answer: A

Explanation: Running OPTIMIZE logs_data ZORDER BY (event, timestamp) compacts small files and applies Z-order indexing on event and timestamp, optimizing data skipping for queries.

Question: 530

A data engineer creates a deep clone of a Delta table, source_employees, to target_employees_clone using CREATE TABLE target_employees_clone DEEP CLONE source_employees. The source table has a check constraint salary > 0. The engineer updates the target table with UPDATE target_employees_clone SET salary = -100 WHERE employee_id = 1. What happens?

A. The update fails because deep clones reference the source table's constraints
B. The update succeeds because deep clones do not inherit check constraints
C. The update succeeds but logs a warning about the constraint violation
D. The update fails because the check constraint is copied to the target table
E. The update requires disabling the constraint on the target table first

Answer: D

Explanation: A deep clone copies both data and metadata, including check constraints like salary > 0. The UPDATE operation on the target table (target_employees_clone) violates this constraint, causing the operation to fail. Deep clones are independent, so constraints are not referenced from the source but are enforced on the target. No warnings are logged, and disabling constraints is not required unless explicitly done.

Question: 531

A dynamic view on the Delta table employee_data (emp_id, name, salary, dept) must mask salary as NULL for non-hr members and restrict rows to dept = 'HR' for non-manager members. The view must be optimized for a Unity Catalog-enabled workspace. Which SQL statement is correct?

A. CREATE VIEW emp_view AS SELECT emp_id, WHEN is_member('hr') THEN salary ELSE NULL END AS salary, name, dept FROM employee_data WHERE is_member('manager') OR dept = 'HR';
B. CREATE VIEW emp_view AS SELECT emp_id, IF(is_member('hr'), NULL, salary) AS salary, name, dept FROM employee_data WHERE dept = 'HR' OR is_member('manager');
C. CREATE VIEW emp_view AS SELECT emp_id, MASK(salary, 'hr') AS salary, name, dept FROM employee_data WHERE dept = 'HR' AND NOT is_member('manager');
D. CREATE VIEW emp_view AS SELECT emp_id, COALESCE(is_member('hr'), salary, NULL) AS salary, name, dept FROM employee_data WHERE dept = 'HR';
E. CREATE VIEW emp_view AS SELECT emp_id, CASE WHEN is_member('hr') THEN salary ELSE NULL END AS salary, name, dept FROM employee_data WHERE

CASE WHEN is_member('manager') THEN TRUE ELSE dept = 'HR' END;

Answer: E

Explanation: The view must mask salary and restrict rows in a Unity Catalog-enabled workspace. The first option uses CASE statements correctly. The second option reverses the masking logic. The third option uses a non-existent MASK function. The fourth option misuses COALESCE. The fifth option has a syntax error with WHEN.

Question: 532

A data engineer is deduplicating a streaming DataFrame events with columns event_id, user_id, and timestamp. Duplicates occur within a 20-minute window. The deduplicated stream should be written to a Delta table events_deduped in append mode. Which code is correct?

A. events.dropDuplicates("event_id").withWatermark("timestamp", "20 minutes").writeStream.outputMode("append").table("events_deduped")
B. events.withWatermark("timestamp", "20 minutes").dropDuplicates("event_id").writeStream.outputMode("append").table("events_deduped")
C. events.withWatermark("timestamp", "20 minutes").groupBy("event_id").agg(max("timestamp")).writeStream.outputMode("complete").table
D. events.dropDuplicates("event_id", "timestamp").writeStream.outputMode("append").table("events_deduped")
E. events.withWatermark("timestamp", "20 minutes").distinct().writeStream.outputMode("update").table("events_deduped")

Answer: B

Explanation: Deduplication requires withWatermark("timestamp", "20 minutes") followed by dropDuplicates("event_id") to remove duplicates within the 20-minute window. The append mode writes deduplicated records to the Delta table.

Question: 533

A Databricks job failed in Task 5 due to a data quality issue in a Delta table. The task uses a Python file importing a Wheel-based module quality_checks. The team refactors to use /Repos/project/checks/quality_checks.py. How should the engineer repair the task

and refactor the import?

A. Run OPTIMIZE, rerun the job, and import using import sys; sys.path.append("/Repos/project/checks")
B. Use FSCK REPAIR TABLE, repair Task 5, and import using from checks.quality_checks import *
C. Delete the Delta table, rerun Task 5, and import using from /Repos/project/checks/quality_checks import *
D. Use the Jobs API to reset the job, and import using from ..checks.quality_checks import *
E. Clone the job, increase cluster size, and import using from checks import quality_checks

Answer: B

Explanation: Using FSCK REPAIR TABLE addresses data quality issues in the Delta table, and repairing Task 5 via the UI targets the failure. The correct import is from checks.quality_checks import *. Running OPTIMIZE doesn't fix data quality. Deleting the table causes data loss. Resetting or cloning the job is unnecessary. Double-dot or incorrect package imports fail.

Question: 534

A data engineer is implementing a streaming pipeline that processes IoT data with columns device_id, timestamp, and value. The pipeline must detect anomalies where value exceeds 100 for more than 5 minutes. Which code block achieves this?

A. df = spark.readStream.table("iot_data") \
.withWatermark("timestamp", "5 minutes") \
.groupBy("device_id", window("timestamp", "5 minutes")) \
.agg(max("value").alias("max_value")) \
.filter("max_value > 100") \
.writeStream \
.outputMode("update") \
.start()
B. df = spark.readStream.table("iot_data") \
.withWatermark("timestamp", "5 minutes") \
.groupBy("device_id", window("timestamp", "5 minutes")) \
.agg(max("value").alias("max_value")) \

```
.filter("max_value > 100") \
.writeStream \
.outputMode("append") \
.start()
```
C.
```
df = spark.readStream.table("iot_data") \
.groupBy("device_id", window("timestamp", "5 minutes")) \
.agg(max("value").alias("max_value")) \
.filter("max_value > 100") \
.writeStream \
.outputMode("complete") \
.start()
```
D.
```
df = spark.readStream.table("iot_data") \
.withWatermark("timestamp", "5 minutes") \
.filter("value > 100") \
.groupBy("device_id", window("timestamp", "5 minutes")) \
.count() \
.writeStream \
.outputMode("append") \
.start()
```

Answer: A

Explanation: Detecting anomalies requires aggregating max(value) over a 5-minute window and filtering for max_value > 100. The update mode outputs only updated aggregates, suitable for anomaly detection. append mode is invalid for aggregations. complete mode is inefficient for streaming.

Question: 535

Objective: Evaluate the behavior of a streaming query with watermarking

A streaming query processes a Delta table stream_logs with the following code:

```
spark.readStream
.format("delta")
.table("stream_logs")
.withWatermark("event_time", "10 minutes")
.groupBy(window("event_time", "5 minutes"))
.count()
```
If a late event arrives 15 minutes after its event_time, what happens?

A. The event is included in the current window and processed.
B. The event is buffered until the next trigger.
C. The event is processed in a new window.
D. The query fails due to late data.
E. The event is dropped due to the watermark.

Answer: E

Explanation: The withWatermark("event_time", "10 minutes") setting discards events that arrive more than 10 minutes late. A 15-minute-late event is dropped and not included in any window.

Question: 536

A streaming pipeline processes user activity into an SCD Type 2 Delta table with columns user_id, activity, start_date, end_date, and is_current. The stream delivers user_id, activity, and event_timestamp. Which code handles intra-batch duplicates and late data?

A. MERGE INTO activity t USING (SELECT user_id, activity, event_timestamp FROM source WHERE event_timestamp > (SELECT MAX(end_datA. FROM activity)) s ON t.user_id = s.user_id AND t.is_current = true WHEN MATCHED AND t.activity != s.activity THEN UPDATE SET t.is_current = false, t.end_date = s.event_timestamp WHEN NOT MATCHED THEN INSERT (user_id, activity, start_date, end_date, is_current) VALUES (s.user_id, s.activity, s.event_timestamp, null, true)
B. MERGE INTO activity t USING source s ON t.user_id = s.user_id WHEN MATCHED THEN UPDATE SET t.activity = s.activity, t.start_date = s.event_timestamp WHEN NOT MATCHED THEN INSERT (user_id, activity, start_date, end_date, is_current) VALUES (s.user_id, s.activity, s.event_timestamp, null, true)
C. spark.readStream.table("source").writeStream.format("delta").option("checkpointLocation", "/checkpoints/activity").outputMode("append").table("activity")
D. spark.readStream.table("source").groupBy("user_id").agg(max("activity").alias("activity"), max("event_timestamp").alias("start_date")).writeStream.format("delta").option("checkpointLocati "/checkpoints/activity").outputMode("complete").table("activity")
E. spark.readStream.table("source").withWatermark("event_timestamp", "30 minutes").dropDuplicates("user_id", "event_timestamp").writeStream.format("delta").option("checkpointLocation", "/checkpoints/activity").outputMode("append").table("activity")

Answer: A

Explanation: SCD Type 2 requires maintaining historical records, and streaming pipelines must handle intra-batch duplicates and late data. The MERGE operation filters source records to include only those with event_timestamp greater than the maximum end_date, ensuring late data is processed correctly. It matches on user_id and is_current, updating the current record to inactive and setting end_date if the activity differs, then inserts new records. Watermarking with dropDuplicates alone risks losing history, append mode without MERGE does not handle updates, and complete mode is inefficient. A simple MERGE without timestamp filtering mishandles late data.

Question: 537

A data engineer is tasked with securing a Delta table sensitive_data containing personally identifiable information (PII). The table must be accessible only to users in the data_analysts group with SELECT privileges, and all operations must be logged. Which combination of SQL commands achieves this?

A. GRANT SELECT ON TABLE sensitive_data TO data_analysts;
SET TBLPROPERTIES ('delta.enableChangeDataFeed' = 'true');
B. GRANT SELECT ON TABLE sensitive_data TO data_analysts;
ALTER TABLE sensitive_data SET TBLPROPERTIES ('delta.enableAuditLog' = 'true');
C. GRANT READ ON TABLE sensitive_data TO data_analysts;
ALTER TABLE sensitive_data ENABLE AUDIT LOG;
D. GRANT SELECT ON TABLE sensitive_data TO data_analysts;
ALTER TABLE sensitive_data SET TBLPROPERTIES ('audit_log' = 'true');

Answer: B

Explanation: GRANT SELECT assigns read-only access to the data_analysts group. Enabling audit logging requires setting the Delta table property delta.enableAuditLog to true using ALTER TABLE ... SET TBLPROPERTIES.

Question: 538

A Delta Lake table transactions has columns tx_id, account_id, and amount. The team wants to ensure amount is not null and greater than 0. Which command enforces this?

A. ALTER TABLE transactions ADD CONSTRAINT positive_amount CHECK (amount > 0 AND amount IS NOT NULL)
B. ALTER TABLE transactions MODIFY amount NOT NULL, ADD CONSTRAINT positive_amount CHECK (amount > 0)
C. ALTER TABLE transactions SET amount NOT NULL, CONSTRAINT positive_amount CHECK (amount > 0)
D. CREATE CONSTRAINT positive_amount ON transactions CHECK (amount > 0 AND amount IS NOT NULL)
E. ALTER TABLE transactions MODIFY amount CHECK (amount > 0) NOT NULL

Answer: B

Explanation: The correct syntax is ALTER TABLE transactions MODIFY amount NOT NULL, ADD CONSTRAINT positive_amount CHECK (amount > 0), applying both constraints separately.

Question: 539

A data engineering team is automating cluster management using the Databricks CLI. They need to create a cluster with 4 workers, a specific runtime (13.3.x-scala2.12), and auto-termination after 60 minutes. The command must use a profile named AUTO_PROFILE. Which command correctly creates this cluster?

A. databricks clusters create --profile AUTO_PROFILE --name auto-cluster --workers 4 --runtime 13.3.x-scala2.12 --auto-terminate 60
B. databricks clusters create --json '{"cluster_name": "auto-cluster", "num_workers": 4, "spark_version": "13.3.x-scala2.12", "autotermination_minutes": 60}' --profile AUTO_PROFILE
C. databricks clusters start --json '{"cluster_name": "auto-cluster", "num_workers": 4, "spark_version": "13.3.x-scala2.12", "autotermination_minutes": 60}' --profile AUTO_PROFILE
D. databricks clusters create --profile AUTO_PROFILE --cluster-name auto-cluster --num-workers 4 --version 13.3.x-scala2.12 --terminate-after 60
E. databricks clusters configure --profile AUTO_PROFILE --cluster auto-cluster --workers 4 --spark-version 13.3.x-scala2.12 --auto-termination 60

Answer: B

Explanation: The databricks clusters create command requires a JSON specification for cluster configuration when using the --json flag. The correct command is databricks clusters create --json '{"cluster_name": "auto-cluster", "num_workers": 4, "spark_version": "13.3.x-scala2.12", "autotermination_minutes": 60}' --profile AUTO_PROFILE. This specifies the cluster name, number of workers, Spark runtime version, and auto-termination period. Other options are incorrect: B and D use invalid flags (--workers, --runtime, --name, --version, --terminate-after); C uses start instead of create, which is for existing clusters; E uses an invalid configure command.

Question: 540

A data engineer needs to import a notebook from a local file (/local/notebook.py) to a workspace path (/Users/user/new_notebook) using the CLI with profile IMPORT_PROFILE. Which command achieves this?

A. databricks workspace copy /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE
B. databricks notebook import /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE
C. databricks workspace upload /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE
D. databricks workspace import /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE
E. databricks notebook push /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE

Answer: D

Explanation: The databricks workspace import command imports a local file to a workspace path. The correct command is databricks workspace import /local/notebook.py /Users/user/new_notebook --profile IMPORT_PROFILE. Other options are incorrect: B, D, and E use invalid commands (notebook import, workspace copy, notebook push); C uses an invalid workspace upload command.

Question: 541

A streaming pipeline propagates deletes from a Delta table orders to orders_history using CDF. The pipeline fails due to high latency during peak hours. Which configuration

improves performance?

A. Run OPTIMIZE orders ZORDER BY order_id daily
B. Use spark.readStream.option("maxFilesPerTrigger", 1000).table("orders")
C. Increase spark.sql.shuffle.partitions to 1000
D. Set spark.databricks.delta.optimize.maxFileSize = 512MB
E. Disable CDF and use a batch MERGE INTO operation

Answer: B

Explanation: High latency in a CDF streaming pipeline during peak hours can result from processing too many files. Setting spark.readStream.option("maxFilesPerTrigger", 1000) limits the number of files processed per micro-batch, controlling latency. OPTIMIZE helps batch performance but not streaming, maxFileSize requires OPTIMIZE, increasing shuffle partitions increases overhead, and disabling CDF defeats the purpose.

KILL EXAMS
KILL EXAMS
KILL EXAMS

KILLEXAMS.COM

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including Exam Questions, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.

Practice Exam Questions Based on Current Exam Objectives

Killexams.com provides practice exam questions aligned with the latest official exam objectives and latest syllabus. Our content is reviewed and updated regularly to reflect recent changes announced by certification vendors. By studying these practice questions, candidates will cover the structure, difficulty level, and topics of the actual exam, helping them prepare more effectively and efficiently.

Comprehensive Practice Exams (PDF Format)

Killexams.com offers multiple-choice questions (MCQs) in easy-to-read PDF format, covering all major domains of the exam. Each PDF contains a structured collection of practice questions and verified answers designed to support focused study. These MCQs help candidates reinforce key concepts, identify knowledge gaps, and improve exam readiness through consistent practice.

Realistic Practice Tests (Online Test Engine & Desktop Test Engine)

To support hands-on preparation, Killexams.com provides practice tests through both an Online Test Engine and a Desktop Test Engine. These tools are designed to simulate a real exam environment, allowing candidates to practice under exam-like conditions, with latest syllbus and topics of the exam. Performance tracking, test history, and result analysis help users evaluate their progress and focus on areas that need improvement.

Risk-Free Purchase Policy

Killexams.com follows a transparent and customer-friendly purchase policy. If users are not satisfied with the study materials, they may request assistance or a refund in accordance with our published terms and conditions. This policy reflects our commitment to customer satisfaction, fairness, and confidence in our preparation resources.

Regularly Updated Content

Our practice question bank is reviewed and updated on an ongoing basis to stay aligned with the latest exam outlines and vendor updates. This ensures candidates are studying up-to-date, relevant material, and preparing with content that reflects current exam expectations, helping them stay confident and well-prepared.