CERTLABZ CERTIFICATION BOARD

Certified Database Professional (CDBP)

Ten Sample Questions

These ten questions are written at the difficulty of the live exam. Six carry a figure, a data table, a configuration or a code listing; four are plain scenario questions, as on the live paper. Every one is a scenario a working database professional would meet, and none of them appears in the live question bank.

The live exam is multistage adaptive: how you score on one stage selects the difficulty of the next, so no two candidates sit the same paper. It is a performance-based exam: each form can carry up to 5 to 12 performance-based questions alongside the multiple-choice questions, which means you build, sequence or complete something rather than only choosing from a list. Answers and full explanations follow each question here; the live exam does not disclose item-level answers.

Question 1 — Domain 1: Relational Database Foundations — DATA

Emma Wilson matches each relational constraint to its job.

Constraint	Job
Primary key	(to choose)
Foreign key	(to choose)
Unique constraint	(to choose)
Check constraint	(to choose)

Table 1.1 Constraints and their jobs.

Which set of jobs is correct?

A. Blocks duplicate values, enforces a value rule, links tables and identifies each row, so the primary key ends up merely blocking duplicate column values.

B. Identifies each row uniquely, links to another tables key, blocks duplicate values, and enforces a value rule, matching each constraint to its real job.

C. Links to another table, identifies each row, enforces a value rule and blocks duplicates, so the foreign key is what identifies each row on its own.

D. Enforces a value rule, blocks duplicate values, identifies each row and links tables, so a check constraint is what uniquely identifies every row here.

Correct answer: B
A primary key identifies each row uniquely, a foreign key references a key in another table, a unique constraint blocks duplicate values in a column, and a check constraint enforces a value rule. Only option B pairs each constraint with its actual job.

Question 2 — Domain 1: Relational Database Foundations — CONFIG

James Carter reviews a table that has no key, and several identical rows have been stored.

```
table         : orders
primary key   : none
duplicate rows: present
row identity  : ambiguous
```

Listing 2.1 The current table.

What is the soundest change?

A. Leave the table without a key, since a relational table works perfectly well when several identical rows share every column value at the same time.

B. Add a second copy of every column, on the assumption that widening each row makes duplicate rows far easier for the engine to tell apart later on.

C. Define a primary key on a column or set of columns that is unique and not null, so every row has one identity and duplicate rows cannot be stored.

D. Sort the table by every one of its columns, expecting the ordering alone to remove the duplicate rows and give each remaining row a stable identity.

Correct answer: C
A primary key on a unique, not-null column or set of columns gives every row a single identity and prevents duplicate rows from being stored. Leaving the table keyless allows the duplicates, widening rows does not add identity, and sorting does not remove duplicates.

Question 3 — Domain 2: SQL and Query Design — CONFIG

Olivia Davis reviews a query meant to list each customer beside the orders they placed.

```
SELECT c.name, o.total
FROM customers c, orders o
-- no join condition given
result: every possible pair
```

Listing 3.1 The current query.

What is the soundest fix?

A. Join the tables on the matching key, using customers.id equal to orders.customer_id, so each order pairs only with the customer that placed it.

B. Add a LIMIT clause to the query, on the assumption that returning fewer rows repairs the meaning of a join that has no condition between the tables.

C Order the result by the customer name, expecting the sort to remove the unrelated pairings that the missing join condition has already produced here.

D Select fewer columns in the query, since narrowing the column list is what stops two tables from producing every possible pair of their rows together.

Correct answer: A
Listing two tables with no join condition produces a Cartesian product of every possible pair. Adding the join condition customers.id equal to orders.customer_id pairs each order only with its own customer. A LIMIT, an ORDER BY, or fewer columns do not repair the missing condition.

Question 4 — Domain 2: SQL and Query Design — DATA

Liam Anderson maps each SQL clause to what it does in a grouped query.

Clause	Role
WHERE	(to choose)
GROUP BY	(to choose)
HAVING	(to choose)
ORDER BY	(to choose)

Table 4.1 Clauses and their roles.

Which set of roles is correct?

A Sorts the final result, groups the rows, filters groups after aggregation and filters rows, so WHERE is the clause that sorts the output at the very end.

B Groups rows, filters rows before grouping, sorts the final result and filters groups, so HAVING is what filters individual rows before any grouping happens.

C Filters groups after aggregation, sorts the result, filters rows and groups them, so GROUP BY is the clause that sorts the final result for the reader here.

D Filters rows before grouping, groups rows for aggregation, filters groups after aggregation and sorts the result, matching each clause to its real job here.

Correct answer: D
WHERE filters rows before grouping, GROUP BY groups rows for aggregation, HAVING filters the resulting groups, and ORDER BY sorts the final result. Only option D pairs each clause with what it actually does in a grouped query.

Question 5 — Domain 3: Schema Design and Data Modeling — DATA

Ava Thompson matches each modeling problem to the change that fixes it.

Problem	Change
Same fact repeated in many rows	(to choose)
Repeating columns like phone1, phone2	(to choose)
Two tables in a many-to-many link	(to choose)
Value depends on part of a key	(to choose)

Table 5.1 Modeling problems and changes.

Which set of changes is correct?

A. Add a junction table, split into its own table, move to a related table and move to its own table, so a repeated fact is fixed by adding a junction table.

B. Move the fact to its own table, move repeats to a related table, add a junction table, and split the value out, matching each problem to its real change here.

C. Move to a related table, add a junction table, move to its own table and split out, so a many-to-many link is fixed by moving repeats to another table here.

D. Split into its own table, move a fact to its own table, add a junction and move repeats, so repeating columns are fixed by splitting on the composite key.

Correct answer: B
A fact repeated across rows belongs in its own table, repeating columns move to a related table, a many-to-many link needs a junction table, and a value that depends on part of a key is split into its own table. Only option B pairs each problem with the change that resolves it.

Question 6

Domain 4: Performance and Indexing

CONFIG

David Brown reviews a slow lookup that filters a large table on one column.

```
query         : WHERE email = value
rows in table : 5 million
index on email: none
plan          : full table scan
```

Listing 6.1 The current plan.

What is the soundest change?

A. Rewrite the filter as email LIKE a leading wildcard, on the assumption that a leading-wildcard pattern lets the engine skip the full table scan here.

B. Add more memory to the server only, since extra memory on its own removes the need for any index when a large table is filtered on a single column.

C. Create an index on the email column, so the engine can seek directly to the matching rows instead of scanning all five million rows on each lookup.

D. Select every column with a star, expecting the wider result set to guide the planner toward a faster path than the current full table scan gives it.

Correct answer: C
An index on the filtered email column lets the engine seek directly to matching rows rather than scanning all five million. A leading-wildcard pattern cannot use an index, added memory does not replace one, and selecting more columns does not help the planner avoid the scan.

Question 7 Domain 4: Performance and Indexing

A table carries a separate index on every column, writes have grown slow, and a common two-column filter is still slow. What is the soundest step?

A. Replace redundant single-column indexes with a composite index that matches the common filter, so writes carry less overhead and the filter is served well.

B. Add one more single-column index for every column in the table, since adding still more indexes is what finally makes both the writes and the filter fast.

C. Drop all of the indexes from the table permanently, because a table with no indexes at all is always faster for both reads and writes under any workload here.

D. Raise the isolation level on the connection, on the assumption that stricter isolation is what makes an under-served filter query come back faster than now.

Correct answer: A
A composite index ordered to match the common two-column filter serves that query while letting redundant single-column indexes be removed, which lowers write overhead. Adding more indexes worsens writes, dropping all indexes slows reads, and isolation level does not speed up a filter.

Question 8 Domain 5: Transactions, Concurrency and Integrity

Two transactions read the same account balance, each adds a deposit, and one update overwrites the other so a deposit is lost. What is the soundest fix?

A. Let both writes proceed as they are, because when two transactions update the same row the database always keeps the sum of the two changes safely intact.

B. Run both transactions on separate database servers, on the assumption that using two servers stops either update from overwriting the other one at all.

C. Remove the primary key from the account row, since a row without a key can be written by many transactions at once with no update ever being lost here.

D. Use a higher isolation level or explicit row locking so the second transaction reads the updated balance, and neither deposit is silently overwritten now.

Correct answer: D
This is a lost update: both transactions read the same balance before either writes. A higher isolation level or explicit row locking forces the second transaction to read the committed change, so neither deposit is overwritten. Multiple servers, keeping the sum automatically, and dropping the key do not prevent it.

Question 9 Domain 6: Security, Operations and Reliability

A team takes nightly database backups but has never restored one, and a recovery drill fails when it is finally attempted. What practice best fixes this?

A. Keep taking backups without ever restoring them, because a backup that has been written to disk is proof enough on its own that a full recovery will succeed.

B. Test restores on a schedule and measure the recovery time, so a backup is trusted only after a real restore has proven that it can bring the data back again.

C. Delete the older backups to save disk space, on the assumption that keeping only the single newest file is what guarantees a recovery will succeed when needed.

D. Store every backup on the same disk as the database, since keeping the backup right beside the live data is what makes a full restore complete more quickly here.

Correct answer: B
A backup is only trustworthy once a real restore has proven it recovers the data, so restores should be tested on a schedule with recovery time measured. Writing files to disk proves nothing, deleting history removes recovery points, and colocating backups with the database risks losing both together.

Question 10 Domain 3: Schema Design and Data Modeling

A team must choose between a relational database and a document store for a new feature. Which reasoning is soundest?

A. Always pick a document store for every new feature, because a schema-free store is faster and simpler than any relational database in every case there is.

B. Always pick a relational database for each new feature, because relational systems handle every possible access pattern better than any other store made.

C. Choose by the data shape and access pattern, using relational for related structured data with strong integrity and a document store for flexible reads.

D. Choose whichever store the newest blog posts praise the most, since following the most recent trend is the surest way to pick the right data store for it.

Correct answer: C
The choice should follow the data shape and access patterns: relational databases suit structured, related data that needs joins and strong integrity, while a document store suits flexible, denormalized, read-heavy access. A blanket rule or the latest trend ignores the actual requirements of the feature.

Published by the CertLabz Certification Board, a division of CertLabz LLC. Companion to objectives document CDBP-001. https://certlabz.com/certifications/certified-database-professional.html