CERTLABZ CERTIFICATION BOARD

Certified SQL Programmer (CSQP)

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 SQL programmer 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 Model and SQL Basics

DATA

Emma Wilson maps each key type to its purpose, with one purpose missing.

Key type	Purpose
Primary key	Unique row identity
Foreign key	(to choose)
Unique key	No duplicate values
Composite key	Two or more columns

Table 1.1 Key types and purpose.

What is the purpose of a foreign key?

A. Sorts the result set
B. Indexes each column
C. Links two tables
D. Removes duplicate rows

Correct answer: C

A foreign key stores a value that references a row in another table, so it links two tables and enforces referential integrity. Sorting, indexing, and removing duplicates are separate concerns handled by ORDER BY, indexes, and unique constraints.

Question 2 — Domain 2: Querying, Filtering and Sorting — CONFIG

James Carter runs this query against a products table.

```
SELECT sku FROM products
WHERE sku LIKE 'A_2%';
```

Listing 2.1 A pattern match query.

Which SKU matches the pattern?

A. A2ZB
B. AX2B
C. BX2A
D. AB3X

Correct answer: B
The pattern A_2% means an A, then any single character, then a 2, then anything, so AX2B matches. A2ZB has no 2 in the third position, BX2A starts with B, and AB3X has a 3 where the 2 must be.

Question 3 — Domain 3: Joins and Subqueries — CONFIG

Olivia Davis needs every customer listed, including those with no orders.

```
SELECT c.name, o.id
FROM customers c
____ orders o
ON c.id = o.cust_id;
```

Listing 3.1 A join with a blank.

Which join keeps all customers?

A. CROSS JOIN
B. INNER JOIN
C. FULL JOIN
D. LEFT JOIN

Correct answer: D
A left join returns every row from the left table, customers, and fills nulls where no order matches. An inner join drops unmatched customers, a cross join pairs every combination, and a full join would also add orphan orders.

Question 4 — Domain 4: Aggregation and Grouping — DATA

Ava Thompson counts orders per customer and wants only customers with more than three orders.

Customer	Orders
Miller	5
Carter	2
Brown	4
Davis	1

Table 4.1 Order counts per customer.

Which clause filters these grouped counts?

A WHERE
B ORDER BY
C HAVING
D GROUP BY

Correct answer: C
HAVING filters aggregated groups after GROUP BY runs, so it is the clause that keeps only customers with a count above three. WHERE filters individual rows before grouping and cannot reference an aggregate.

Question 5

Domain 5: Data Definition and Modification

CONFIG

Liam Anderson defines a table so that each email can appear only once.

```
CREATE TABLE members (
  id INT PRIMARY KEY,
  email VARCHAR(100) ____
);
```

Listing 5.1 A table with a blank constraint.

Which constraint enforces this?

A CHECK
B NOT NULL
C DEFAULT
D UNIQUE

Correct answer: D
A UNIQUE constraint prevents any value in the email column from repeating, which is exactly the rule described. NOT NULL only forbids empty values, CHECK validates a condition, and DEFAULT supplies a fallback value.

Question 6 Domain 6: Indexes, Transactions and Performance DATA

David Brown compares isolation levels and the anomaly each still allows, with one entry missing.

Isolation level	Still allows
Read uncommitted	Dirty reads
Read committed	(to choose)
Repeatable read	Phantom reads
Serializable	None

Table 6.1 Isolation levels and anomalies.

What does read committed still allow?

A. Every anomaly listed
B. Nonrepeatable reads
C. Dirty reads
D. No anomalies at all

Correct answer: B
Read committed stops dirty reads but still permits nonrepeatable reads, where a row reread inside the same transaction returns a changed value. Repeatable read closes that gap, and serializable prevents every anomaly shown.

Question 7 Domain 1: Relational Model and SQL Basics

A bonus column allows NULL, and a query filters with WHERE bonus = 0, but rows where bonus is NULL never appear even though you expect them. Why are the NULL rows excluded?

A. NULL is unknown
B. NULL equals zero
C. WHERE ignores columns
D. Zero is invalid

Correct answer: A
Any comparison against NULL returns unknown rather than true, so those rows fail the WHERE test and drop out. To match them you must use IS NULL, since NULL is never equal to zero or to any other value.

Question 8 Domain 3: Joins and Subqueries

A report needs each employee shown next to their manager, where both are stored as rows in the same employees table. What kind of query retrieves this in one pass?

A. A cross join

B A union query

C A full scan

D A self join

Correct answer: D
A self join joins a table to itself using two aliases, matching each employee row to the manager row in the same table. A cross join pairs every row with every other, and a union stacks separate result sets instead.

Question 9 Domain 4: Aggregation and Grouping

A query mixes COUNT with several plain columns in the SELECT list but has no GROUP BY, and the database rejects it. What rule is being broken?

A Group the columns

B Count needs a join

C Order every column

D Alias each total

Correct answer: A
When a query combines an aggregate such as COUNT with plain columns, every non-aggregated column in the SELECT must appear in GROUP BY. Adding those columns to a GROUP BY clause satisfies the rule and clears the error.

Question 10 Domain 6: Indexes, Transactions and Performance

A frequently run query filters on a large unindexed column and does a full table scan every time, so it grows slower as the table grows. What is the standard fix?

A Drop the table

B Remove the WHERE

C Add an index

D Run it twice

Correct answer: C
Adding an index on the filtered column lets the database seek the matching rows instead of scanning the whole table, which keeps the query fast as data grows. Dropping the table or removing the filter would not serve the query at all.

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