1. A many-to-many relationship becomes –

   ☐ a foreign key column

   ☐ a view

   ☐ nothing

   ☐ its own junction table

2. The operator that keeps columns is –

   ☐ selection

   ☐ union

   ☐ join

   ☐ projection

3. Changing a course title in 14 enrollment rows is –

   ☐ an update anomaly

   ☐ correct behaviour

   ☐ an insert anomaly

   ☐ a delete anomaly

4. A fan-out-100 B-tree of height 3 covers – keys

   ☐ 10 000

   ☐ 100 million

   ☐ 300

   ☐ 1 000 000

5. The A in ACID stands for –

   ☐ availability

   ☐ audit

   ☐ atomicity

   ☐ async

6. Authorisation decides –

   ☐ what you may do

   ☐ who you are

   ☐ how fast queries run

7. PRIMARY KEY (student, course, term) enforces –

   ☐ one enrollment per student-course-term

   ☐ nothing without triggers

   ☐ one enrollment per course

   ☐ one enrollment per student

8. Filtering on COUNT(*) happens in –

   ☐ ON

   ☐ HAVING

   ☐ WHERE

   ☐ GROUP BY

9. A part-key dependency breaks –

   ☐ 3NF

   ☐ BCNF only

   ☐ 1NF

   ☐ 2NF

10. EXPLAIN QUERY PLAN without an index prints –

    ☐ LOCK

    ☐ SCAN

    ☐ nothing

    ☐ SEARCH USING INDEX

1. **its own junction table**

   The junction table holds the foreign-key pairs; neither end can hold the other.

2. **projection**

   Projection pi keeps the chosen columns; selection sigma keeps rows.

3. **an update anomaly**

   One fact stored many times: the update anomaly.

4. **1 000 000**

   m to the h: 100 cubed is a million - 3 reads per lookup.

5. **atomicity**

   All or nothing: the transaction commits completely or not at all.

6. **what you may do**

   Authentication proves identity; authorisation bounds the rights.

7. **one enrollment per student-course-term**

   The composite key rejects duplicate triples at the engine level.

8. **HAVING**

   WHERE filters rows before grouping; HAVING filters the groups.

9. **2NF**

   2NF removes dependencies on part of the key; 3NF the transitive ones.

10. **SCAN**

    The engine says what it will do: SCAN is the full-table read.