Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

Each answer shows the steps, then the **result**.

2.1

Ana

Chen

2.2

CS101: 3 rows, average $217/3 \approx$ **72.3**

MA101: `COUNT(*)` is 2

MA101 average 55: `AVG` skips the NULL

2.3

```
SELECT s.name FROM student s LEFT JOIN enrol e ON s.sid = e.sid
WHERE e.sid IS NULL;
```

Dara

2.4

$\sigma_{\text{credits}>5}(\text{course})$

then $\pi_{\text{title}}(\dots)$

3.1

(a) `SELECT s.name, e.mark FROM student s JOIN enrol e ON s.sid = e.sid WHERE e.cid = 'CS101' ORDER BY e.mark DESC;`; Chen 81, Ana 72, Ben 64

(b) `SELECT s.name, COUNT(e.cid) FROM student s LEFT JOIN enrol e ON s.sid = e.sid GROUP BY s.sid, s.name;`; Ana 2, Ben 1, Chen 2, Dara 0

(c) Dara's joined row exists with NULLs, so `COUNT(*)` would give 1; `COUNT(e.cid)` skips the NULL and gives 0

3.2

(a) **4**, not 5: both comparisons with NULL are unknown, so the row fails `WHERE`

(b) SQL: **55**; as 0: $(55 + 0)/2 =$ **27.5**, which is wrong for a missing mark

(c) `WHERE mark IS NULL`

3.3

(a) e.g.

```
SELECT e.cid, s.name, e.mark
FROM enrol e JOIN student s ON s.sid = e.sid
JOIN (SELECT cid, MAX(mark) AS top FROM enrol GROUP BY cid) t
  ON t.cid = e.cid AND e.mark = t.top;
```


exercises.lol exercises.com.cn

the maximum per course; joined back to find the rows with it; the student's name

(b) both rows appear; add a rule such as `ORDER BY` name with `ROW_NUMBER()` to keep one, or show both as joint top

(c) the subquery may run once per row: about 10^7 times instead of once

(d) it sends 10^7 rows over the network and uses Python memory, where the database could do it with one pass and indexes; do the grouping in SQL and fetch only one row per course