ANSWERS

F7.4 SQL: two tables and changing data

Files and databases · GCSE · OCR J277 2.2.3, AQA 8525 3.7.2 · about 20 min

What this lesson is about

Queries across two tables; INSERT, UPDATE and DELETE, and the robot logging its own runs.

- Questions across two tables
- INSERT: adding a record
- Values from the program
- UPDATE and DELETE

Questions

5 marks in all

1. In `SELECT robots.name, runs.cm FROM runs, robots WHERE runs.robot_id = robots.robot_id`, what does the WHERE condition do? [1 mark]

- ☐ **A** Matches each run to the robot that did it
- ☐ **B** Deletes robots with no runs
- ☐ **C** Sorts the runs
- ☐ **D** Counts the robots

> **Answer:** A. It matches the foreign key in runs to the primary key in robots.

2. Which statement adds a new record? [1 mark]

- ☐ **A** INSERT INTO
- ☐ **B** UPDATE
- ☐ **C** SELECT
- ☐ **D** DELETE FROM

> **Answer:** A. INSERT INTO table (fields) VALUES (values).

3. What does `DELETE FROM runs` do with no WHERE? [1 mark]

- ☐ **A** Deletes every record in runs
- ☐ **B** Deletes nothing
- ☐ **C** Deletes the table's last record
- ☐ **D** Gives an error

> **Answer:** A. Without WHERE, every record matches. Always check the condition.

4. Complete the statement to change Ada's colour to purple: `_____ robots SET colour = 'purple' WHERE name = 'Ada'` [1 mark]

> **Answer:** UPDATE. UPDATE table SET field = value WHERE condition.

5. Why pass values to a query with `?` placeholders instead of joining text into the SQL? [1 mark]

- ☐ **A** It stops SQL injection, where typed text changes what the query does
- ☐ **B** It makes the query shorter
- ☐ **C** SQL cannot contain numbers
- ☐ **D** Placeholders are faster to type

> **Answer:** A. The database treats placeholder values as data, never as SQL.

The task: log and correct

Using `logbook.open_db()`: drive forward 35 cm and **insert** it as run 7 for robot 2 (Bolt), task `wall`, with the distance the robot actually covered (`position()`, rounded to 1 decimal place) and its time. **Update** run 3's task to `ramp`. **Delete** run 5. Then, with one query across both tables, print every run by Bolt, in run order, as `run 3 ramp 38.0`.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import logbook
db = logbook.open_db()
```

The hint students can ask for: Four statements in order: add the new run, change the one with the wrong task, remove the one that should not be there, then a SELECT that joins the two tables to list one robot's runs. Use placeholders for values rather than building the SQL by hand.

A solution

```
{'program': 'from bugbot import *\nconnect()\nimport logbook\ndb =
logbook.open_db()\n\nstart = clock()\nforward(50, distance=35)\nx, y = position()\nseconds
= round(clock() - start, 1)\ndb.execute("INSERT INTO runs VALUES (?, ?, ?, ?, ?)", (7, 2,
"wall", round(y, 1), seconds))\ndb.execute("UPDATE runs SET task = \'ramp\' WHERE run_id =
3")\ndb.execute("DELETE FROM runs WHERE run_id = 5")\nfor run_id, task, cm in
db.execute("""SELECT runs.run_id, runs.task, runs.cm FROM runs, robots\n
WHERE runs.robot_id = robots.robot_id AND robots.name = \'Bolt\'\n
ORDER BY runs.run_id"""):\n    print("run", run_id, task, cm)\n', 'files': {'logbook.py':
'# logbook.py: the class\'s robot runs as a relational database\nimport sqlite3\n\nROBOTS =
[(1, "Ada", "green"), (2, "Bolt", "red"), (3, "Cog", "blue")]\nRUNS = [(1, 1, "wall", 42.0,
3.1), (2, 1, "square", 80.0, 9.4), (3, 2, "wall", 38.0, 2.7),\n        (4, 1, "wall", 45.5,
3.0), (5, 3, "square", 76.5, 8.8), (6, 2, "square", 81.0, 10.2)]\n\ndef open_db():\n
"""A fresh database with the robots and runs tables."""\n    db =
sqlite3.connect(":memory:")\n    db.execute("CREATE TABLE robots (robot_id INTEGER PRIMARY
KEY, name TEXT, colour TEXT)")\n    db.execute("CREATE TABLE runs (run_id INTEGER PRIMARY
KEY, robot_id INTEGER REFERENCES robots(robot_id), "\n                "task TEXT, cm REAL,
seconds REAL)")\n    db.executemany("INSERT INTO robots VALUES (?, ?, ?)", ROBOTS)\n
db.executemany("INSERT INTO runs VALUES (?, ?, ?, ?, ?)", RUNS)\n    return db\n\ndef
show(db, sql):\n    """Run a query and print each record on its own line."""\n    for row
in db.execute(sql):\n        print(*row)\n'}}
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.