ANSWERS

F7.3 SQL: SELECT

Files and databases · GCSE · OCR J277 2.2.3, AQA 8525 3.7.2 · about 15 min

BugBotLab

What this lesson is about

SELECT, FROM, WHERE and ORDER BY on the class's robot runs.

- SELECT and FROM
- WHERE
- ORDER BY
- Using the answers in Python

Questions

5 marks in all

1. Which SQL keyword names the table to search? [1 mark]

- ☐ **A** FROM
- ☐ **B** SELECT
- ☐ **C** WHERE
- ☐ **D** ORDER BY

> **Answer:** A. SELECT names the fields, FROM the table, WHERE the condition.

2. Put the parts of a query in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `ORDER BY cm DESC`
___ `WHERE task = 'wall'`
___ `SELECT run_id, cm`
___ `FROM runs`

> **Answer:**
>
> ```
> SELECT run_id, cm
> FROM runs
> WHERE task = 'wall'
> ORDER BY cm DESC
> ```
>
> SELECT, FROM, WHERE, ORDER BY: always this order.

3. What does `SELECT * FROM robots` return? [1 mark]

- ☐ **A** Every field of every record in robots
- ☐ **B** Only the first record
- ☐ **C** The number of robots
- ☐ **D** Nothing: * is not allowed

> **Answer:** A. * means every field, and with no WHERE every record comes back.

4. Runs have cm values 42, 80, 38, 45.5, 76.5 and 81. Which match `WHERE cm > 76.5`? [1 mark]

Tick every answer that is true.

- ☐ **A** 80
- ☐ **B** 81
- ☐ **C** 76.5
- ☐ **D** 45.5

> **Answer:** A, B. > is strictly greater, so 76.5 itself does not match.

5. Complete the query so it lists the longest runs first: `SELECT run_id, cm FROM runs ORDER BY cm ____` [1 mark]

> **Answer:** DESC. ORDER BY field DESC sorts descending; ASC, the default, ascending.

The task: the longest square runs

Write one SQL query that finds the `run_id` and `cm` of every **square** run longer than **77** cm, longest first, and print each result as `run 6: 81.0 cm`. Then drive forward the distance of the **shortest** of those runs, divided by 4.

```
# 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: One SELECT does the whole job: choose the columns you need, filter to the task and the distance in the WHERE, and order by the distance so the longest is first. The robot then drives from the last row you printed.

A solution

```
{'program': 'from bugbot import *\nconnect()\nimport logbook\ndb =
logbook.open_db()\n\nrows = db.execute("SELECT run_id, cm FROM runs WHERE task = \'square\'
AND cm > 77 ORDER BY cm DESC").fetchall()\nfor run_id, cm in rows:\n    print(f"run
{run_id}: {cm} cm")\nforward(50, distance=rows[-1][1] / 4)\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.