Quiz 11

Databases

10 June 2026

Full name: .....

You have 10 minutes to answer this quiz, directly on this sheet of paper. No electronic devices or material of any kind is allowed. Do not forget to add your name above. Every question amounts to 2 points; the quiz is graded out of 10 points.

Questions

Q1. Circle the only correct answer.

A Flask view handles a successful `POST` to `/book` that inserts a new reservation in the database. What should the view return?

a. the rendered HTML of the confirmation page directly.

b. a JSON object `{"ok": true}` with status `200 OK`.

c. a `303 See Other` redirect (via `redirect(url_for(...))`) to a `GET` URL that displays the result.

d. the same form again, with the previously-submitted values pre-filled.

Q2. Circle all correct answers (there may be more than one).

The following view is intended to delete a reservation:

```python
@app.route("/reservations/<int:id>/delete")
def delete(id):
    db.session.execute(
        f"DELETE FROM reservation WHERE id = {id}"
    )
    db.session.commit()
    return "Deleted."
```

Which of these are real defects of this code?

a. the route is reachable via `GET`, so any browser prefetch, link preview or crawler can delete data.

b. the SQL is built by string interpolation; even though `id` is typed as `int` here, this is the habit that leads to SQL injection in real code.

c. the view does not redirect after the write, so reloading the page would attempt to delete again (and confuse the browser history).

d. the view forgets to call `db.session.add()` before `commit()`.

Q3. Consider the Flask snippet

```
@app.route("/hello")
def hello():
    name = request.args.get("name", "")
    return f"<h1>Hello, {name}!</h1>"
```

Explain briefly what is wrong with returning a string built this way, and propose a correct version.

. . . . . .

. . . . . .

. . . . . .

Q4. Circle the only correct answer.

In a Flask application using SQLAlchemy, what is the recommended lifecycle for a database connection?

a. a single global connection shared by every request of the application.

b. a new connection opened and closed inside every SQL statement.

c. one connection per HTTP request, opened lazily on first use and closed on teardown (`teardown_appcontext`).

d. one connection per Flask view function, regardless of how many requests are served by that view.

Q5. Circle the only correct answer.

Inside a Jinja template, the expression `{{ user.bio|safe }}` takes a string `bio` stored in the database and inserts it into the page *without* HTML-escaping. Which statement is correct?

a. this is necessary, because otherwise the user's text would not be visible at all.

b. this is necessary whenever `bio` contains accented characters such as "é" or "ç".

c. this is dangerous: a user who stores `<script>...</script>` in `bio` could run JavaScript in every other visitor's browser (XSS).

d. this is dangerous only if the application also runs SQL queries from user input.