# Backend Web Dev — Practice

CSCI 40: Computing for the Web

Fall 2026

**How current LLMs scored on this practice bank** *2026-08-26*

| Model | Reasoning | Score |
|---|---|---|
| claude-haiku-4.5 | low effort | 13/15 (87%) |
| claude-sonnet-5 | low effort | 13/15 (87%) |
| claude-fable-5 | low effort | 11/15 (73%) |
| claude-opus-4.8 | low effort | 12/15 (80%) |
| gpt-5.6-luna | low reasoning | 13/15 (87%) |
| gpt-5.4 | low reasoning | 12/15 (80%) |
| gpt-5.6-terra | low reasoning | 10/15 (67%) |
| gpt-5.5 | low reasoning | 13/15 (87%) |
| gpt-5.6-sol | low reasoning | 15/15 (100%) |
| gemini-3.1-flash-lite | minimal thinking | 10/15 (67%) |
| gemini-3.5-flash-lite | minimal thinking | 11/15 (73%) |
| gemini-3.6-flash | minimal thinking | 9/15 (60%) |
| gemini-3.7-flash | low thinking | 13/15 (87%) |

One independent, tool-free prompt per question; deterministic executable ground truth.

### Initial database state used by every problem

```
Every problem starts with a fresh copy of this SQLite database.
NULL denotes a SQL NULL value. Rows are shown in id order.

Schema:
CREATE TABLE users (
   id INTEGER PRIMARY KEY,
   username TEXT NOT NULL UNIQUE,
   password TEXT NOT NULL,
   age INTEGER
);
CREATE TABLE messages (
   id integer primary key,
   sender_id integer not null,
   message text not null,
   created_at timestamp not null default current_timestamp
   );

Initial rows:

users(id, username, password, age):
1 | Trump | Trump | 78
2 | Biden | Biden | 81
3 | Evan | correct horse battery staple | 7
4 | Isaac | soccer | 4
5 | Aaron | guaguagua | 3
6 | Aurelia | | 1
7 | Mike | 524euTjrWm6uK2C5iw8mC6aNgX1JI78o | 35
8 | Kristen | Possible-Rich-Absolute-Battle | NULL

messages(id, sender_id, message, created_at):
1 | 1 | I'm a baby | 2021-11-14 14:30:00
2 | 2 | I'm a baby | 2021-11-14 14:30:00
3 | 3 | I'm a baby | 2021-11-14 14:33:01
4 | 4 | I'm a baby | 2021-11-15 14:35:45
5 | 3 | I'm actually a toddler | 2021-11-16 14:35:45
6 | 6 | Today in 1918, the Armistice that effectively ended WWI came into effect. | 2021-11-11
    11:00:00
7 | 6 | I'm an adult | 2021-11-17 14:35:45
8 | 6 | SQL is the best!! | 2021-11-17 14:35:45
9 | 7 | I'm an adult | 2021-11-17 14:35:45
```

```
10 | 7 | WTF is SQL?! I thought you liked the snake thing. | 2021-11-17 14:35:45
```

**Quiz rules:**

1. You MAY use any printed or handwritten notes.
2. You MAY NOT use a computer or any other electronic device.

For each program below, write the output of all print statements. You MAY NOT use a computer. **problem1.py**

```python
# Problem 1

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()
sql = """
SELECT id,password FROM users WHERE username='Trump';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem1b.py**

```python
# Problem 1b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()
sql = """
SELECT count(*) FROM messages WHERE message LIKE '%a%';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
```

**problem2.py**

```python
# Problem 2

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

sql = """
INSERT INTO users (username, password) VALUES ('example', 'password');
"""
cur.execute(sql)
con.commit()

sql = """
SELECT id,password FROM users WHERE username='example';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem2b.py**

```
# Problem 2b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

sql = """
INSERT INTO users (username, password, age) VALUES ('example', 'password', 22);
"""
cur.execute(sql)
con.commit()

sql = """
SELECT password,id FROM users WHERE username='example';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem3.py**

```
# Problem 3

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = 'example'
password = 'password'
sql = """
INSERT INTO users (username, password) VALUES ('"""+username+"""', '"""+password+"""');
"""
cur.execute(sql)
con.commit()

sql = """
SELECT id,password FROM users WHERE username='example';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem3b.py**

```
# Problem 3b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = "example', 'password'), ('example2"
password = 'password'
sql = """
INSERT INTO users (username, password) VALUES ('"""+username+"""', '"""+password+"""');
"""
cur.execute(sql)
con.commit()

sql = """
```

```
SELECT id,password FROM users WHERE username LIKE 'example%';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem4.py**

```
# Problem 4

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = 'example'
password = 'password'
sql = """
INSERT INTO users (username, password) VALUES (?, ?);
"""
cur.execute(sql, [username, password])
con.commit()

sql = """
SELECT id,password FROM users WHERE username='example';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem4b.py**

```
# Problem 4b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = "example', 'password'), ('example2"
password = 'password'
sql = """
INSERT INTO users (username, password) VALUES (?, ?);
"""
cur.execute(sql, [username, password])
con.commit()

sql = """
SELECT id,password FROM users WHERE username='example';
"""
cur.execute(sql)
for row in cur.fetchall():
  print('row[0]=', row[0])
  print('row[1]=', row[1])
```

**problem4c.py**

```
# Problem 4b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()
```

```
7
8  username = "example"
9  password = 'password'
10 sql = """
11 INSERT INTO users (username, password) VALUES ('?', ?);
12 """
13 cur.execute(sql, [username])
14 con.commit()
15
16 sql = """
17 SELECT id,password FROM users WHERE username='?';
18 """
19 cur.execute(sql)
20 for row in cur.fetchall():
21   print('row[0]=', row[0])
22   print('row[1]=', row[1])
```

**problem5.py**

```
1  # Problem 5
2
3  import sqlite3
4
5  con = sqlite3.connect('twitter_clone.db')
6  cur = con.cursor()
7
8  username = 'Mike'
9  sql = """
10 SELECT id,password FROM users WHERE username=?;
11 """
12 cur.execute(sql, [username])
13 user_ids = []
14 for row in cur.fetchall():
15    user_ids.append(row[0])
16
17 for user_id in user_ids:
18    sql = """
19    SELECT count(*) FROM messages WHERE sender_id=?;
20    """
21    cur.execute(sql, [user_id])
22    for row in cur.fetchall():
23       print('row[0]=', row[0])
```

**problem5b.py**

```
1  # Problem 5b
2
3  import sqlite3
4
5  con = sqlite3.connect('twitter_clone.db')
6  cur = con.cursor()
7
8  username = "Mike' OR username='Trump"
9  sql = """
10 SELECT id,password FROM users WHERE username='"""+ username +"""';
11 """
12 cur.execute(sql)
13 user_ids = []
14 for row in cur.fetchall():
15    user_ids.append(row[0])
16
17 for user_id in user_ids:
18    sql = """
19    SELECT count(*) FROM messages WHERE sender_id=?;
20    """
21    cur.execute(sql, [user_id])
```

```
   for row in cur.fetchall():
      print('row[0]=', row[0])
```

**problem5c.py**

```
# Problem 5b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = "Mike' OR username='Trump"
sql = """
SELECT id,password FROM users WHERE username=?;
"""
cur.execute(sql, [username])
user_ids = []
for row in cur.fetchall():
   user_ids.append(row[0])

for user_id in user_ids:
   sql = """
   SELECT count(*) FROM messages WHERE sender_id=?;
   """
   cur.execute(sql, [user_id])
   for row in cur.fetchall():
      print('row[0]=', row[0])
```

**problem6.py**

```
# Problem 6b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = 'Mike'
sql = """
SELECT id FROM users WHERE username='""" + username + """';
"""
cur.execute(sql)
user_ids = []
for row in cur.fetchall():
   user_ids.append(row[0])

for user_id in user_ids:
   sql = """
   SELECT id, sender_id, message, created_at FROM messages WHERE sender_id="""" + str(user_id) + "
       "";
   """
   cur.execute(sql)
   for row in cur.fetchall():
      print('id=', row[0], 'sender_id=', row[1], 'message=', row[2], 'created_at=', row[3])
```

**problem6b.py**

```
# Problem 6b

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

```

```
username = 'Mike'
sql = """
SELECT id FROM users WHERE username=?;
"""
cur.execute(sql, [username])
user_ids = []
for row in cur.fetchall():
   user_ids.append(row[0])

for user_id in user_ids:
   sql = """
   SELECT id, sender_id, message, created_at FROM messages WHERE sender_id=?;
   """
   cur.execute(sql, [user_id])
   for row in cur.fetchall():
      print('id=', row[0], 'sender_id=', row[1], 'message=', row[2], 'created_at=', row[3])
```

**problem6c.py**

```
# Problem 6c

import sqlite3

con = sqlite3.connect('twitter_clone.db')
cur = con.cursor()

username = "Mike' OR username='Trump"
sql = """
SELECT id FROM users WHERE username='""" + username + """';
"""
cur.execute(sql)
user_ids = []
for row in cur.fetchall():
   user_ids.append(row[0])

for user_id in user_ids:
   sql = """
   SELECT id, sender_id, message, created_at FROM messages WHERE sender_id="""" + str(user_id) + "
       "";
   """
   cur.execute(sql)
   for row in cur.fetchall():
      print('id=', row[0], 'sender_id=', row[1], 'message=', row[2], 'created_at=', row[3])
```