# Solution

## Solution 6a

No, Alice does not have the correct privileges. She does not need to read the password in the Users table, she does not need INSERT privileges on UserStatus, and she does not need SELECT privileges on LogBook. In addition, Alice lacks INSERT privileges on LogBook.

The minimally required set of permissions is:

```
GRANT SELECT(id, name) ON Users TO Alice;
GRANT SELECT(id, loggedin) ON UserStatus TO Alice;
GRANT INSERT(id, timestamp, name) ON LogBook TO Alice;
```

## Solution 6b

The attacker can input something like `‘ or ‘1‘=‘1`, so that the query turns into

```
SELECT      FROM UserStatus WHERE id = ‘‘ or ‘1‘=‘1‘    ;
```

(mind the closing quote).

This is an example of an SQL injection vulnerability or attack.

The vulnerability can be removed by either correctly sanitizing or escaping the data in the `userinput` variable. A better solution is to use a `PreparedStatement` with placeholder:

```
...
String query = "SELECT * FROM UserStatus WHERE id = ?";
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setString(1, userinput);
ResultSet rs = stmt.executeQuery();
...
```

## Solution 6c

The transaction is vulnerable to “non-repeatable read” and “phantom read” interferences, because the `READ COMMITTED` transaction isolation level does not protect against them. The stronger `REPEATABLE READ` isolation level is not sufficient because it still allows phantom reads. Only the `SERIALIZABLE` isolation level is sufficient, since it protects against dirty read, non-repeatable read and phantom read.