**Name**: _____ **Roll No.**: _____

# CS 349: Lab Midsem: Spring 2025

| Question | Points | Score |
|---|---|---|
| 1 | 11 | |
| 2 | 4 | |
| 3 | 8 | |
| 4 | 16 | |
| 5 | 11 | |
| Total: | 50 | |

1. SQL: Consider the schema $r(\underline{A}, B), s(B, C)$ where $r.A$ is a primary key for $r$.

   (a) (3 points) Suppose we need to find how many $s$ tuples match each $r$ tuple, with value 0 if there is no matching $s$ tuple. Fill in the blanks to create an SQL query for the task; do not use subqueries.

   ```
   SELECT r.A, ____________________________

   FROM r _______________ s ON (_____________)

   GROUP BY _________________
   ```

   **Answer:** COALESCE(COUNT(s.B),0); LEFT OUTER JOIN ; r.B=s.B ; r.A.
   Alternative answer uses COUNT( CASE WHEN s.B IS NULL 0 ELSE 1) instead of coalesce. Using s.C instead of s.B here is wrong since s.C could be null even if s.B is not null, and hence will not get any marks.
   GROUP BY can be r.A, r.B instead of r.A, which will make no difference since r.A is a primary key, but you really should not be adding it. Adding any attribute from s to the GROUP BY will give a wrong answer.
   Rubric: Coalesce: 1, count: 0.5, each other blank 0.5

   (b) (2 points) Answer the same question as above, but using subqueries to fill in the blanks below.

   ```
   SELECT r.A, ____________________________________________________________

   FROM r
   ```

   **Answer:** `(SELECT COUNT(*) FROM s WHERE s.B=r.B|`
   `Rubric: COUNT(*) 1 mark, rest is 1 mark`

   (c) (3 points) Give delete statements to delete all tuples (but not drop the table) from the following relations of the university schema, keeping foreign key dependencies in mind. Explain your answer.
   Relations: `department, instructor, teaches, student, takes, course, section`
   **Answer**: DELETE FROM xx in any order that does not violate the following partial order:

- teaches before (section, instructor)
- takes before (section, student)
- section before (course)
- (student, instructor, course) before department

Eg.
delete from teaches;
delete from takes;
delete from section;
delete from instructor;
delete from student,
delete from course;
delete from department;

Rubric: 1/2 mark for deletion which respects the partial order.

(d) (3 points) Consider the queries
Q1: SELECT r.A, r.B FROM r LEFT OUTER JOIN s ON (r.B=s.B)
Q2: SELECT r.A, r.B FROM r

Assume $r$ contains only the tuple $(a1, b1)$. Give example data for $s$ as well as the results of Q1 and Q2 on the example data, to show that Q1 and Q2 above are not equivalent.

**Answer**: Any instance which has two tuples with the same B value. Output will have duplicates for Q1 but not for Q2

Rubric: 1 for correct dataset, 1 for correct Q1 result on correct dataset, 1 for correct Q2 result on correct dataset.

2. (a) (2 points) Fill in the blanks to make the code below work properly with Psycopg with default settings for autocommit.

```
def executeQuery(self, query):
     try:
         self.co.execute(query)


         ----------------------------
     except Exception as e:


         ------------------------------
         print(e)
```

**Answer:** self.conn.commit(); self.conn.rollback()

Rubric: 1 mark for each blank, no partial marks

(b) (2 points) Fill in the blanks to get a sequence of column names of the relation reln.

```
self.cursor.execute(f"select * from {reln}")

cols = [__________ for col in ___________________________________]
```

**Answer:** cols[0] ; self.cursor.description

Rubric: 1 mark for each blank, no partial marks

3. Node.js and Async Execution

(a) (2 points) In the code snippet in your node.js assignment with express sessions
```
app.get("/list-products", isAuthenticated, async (req, res) => { ... }
```
explain what the isAuthenticated() function does, including how it detects if the user is authenticated, and what it does when the user is not authenticated?

**Answer**: The isAuthenticated method checks if the user has been authenticated by checking if req.session.userId (or any other attribute of session that is assigned after verifying the login user/password) is set in req.session (e.g. by using if(req.session.userId).

If the user is not logged in, it returns a redirect to the login page.

Rubric: 0.5 marks each for what iit checks, and what happens when user not logged in, 1 mark for checking session attribute is set

(b) (3 points) Consider the following code snippet which is part of a function:
```
const result = await pool.query( "SELECT * FROM Products");
print(result.rows)
```
Suppose that the await keyword is removed from this snippet. Explain intuitively what would happen.

**Answer**: This would lead to printing of a promise which does not have any useful information ("junk"), rather than printing of the query result. This is because the await forces execution to complete before the result is assigned, which would be the actual query result. Otherwise the result of an asynchronous call is a promise object; await waits for the promise to be resolved.

An answer which explains the above without mentioning the word promise is acceptable if properly explained.

Rubric: 1 mark for what is printed, 2 marks for explanation.

(c) (3 points) The function bcrypt.hash() is an asynchronous function in the bcrypt library for node.js. Explain why this function may have been made asynchronous, whereas some functions such as Math.random() are synchronous.

**Answer**: By design the hash() function takes time to execute. To avoid blocking during its execution, it is made asynchronous, whereas math.random() executes very fast and there is no need to make it asynchronous.

Rubric: Idea of time of execution being different: 1.5 mark. Explanation of blocking: 1.5 marks

4. Security

(a) (3 points) Passwords are stored as hashes, rather than as clear text, to avoid leakage. What is the risk of using a basic hash function, without the use of "salt" to compute and store the hash?

**Answer**: In this case an attacker can precompute the hash values for common passwords and directly compare the hashes to find the password.

The use of salt avoids this problem since in each account where the password is stored the hash value would be different, resulting in a different hash value.

Rubric: 2 for attack, 1 for why salt blocks the attack.

(b) (4 points) The function bcrypt.hash() returns a single value on a given parameter password, and bcrypt.compare() takes two parameters, (password, hash). Salt is not passed explicitly anywhere. Explain how salt is handled in the above API.

**Answer**: The result of bcrypt(), which we refer to as hash, actually has two parts, the salt and the actual hash, which can be separated out. The compare extracts the salt, computes the hash on password with the salt, and compares with the actual hash value.

Rubric: Idea of hash result having salt and actual hash value: 2 marks, description of computation: 2 marks.

(c) (3 points) Consider the query

```
await pool.query("SELECT * FROM Users where email = '" + req.body[0] + "'"
```

What is the security risk of such a query? Explain by giving an example input. What is the security attack based on this risk called?

**Answer**: By using an input with a quote, a user can add any query they want to the SELECT query. For example, an input "foo' ; DELETE FROM Users; –" results in the query
SELECT * FROM Users WHERE email='foo'; DELETE FROM Users; – '
This attack is called SQL injection.

Rubric: 1 mark for attack via quote, 1.5 mark for example, 0.5 mark for terminology.

(d) (2 points) Instead of using sessions, suppose we set a cookie userid after verifying the userid/password combination, and use the cookie to authenticate subsequent accesses. What is the security risk with this approach?

**Answer**: The risk is that the user can store a cookie with any userid that they want in the browser, which lets them impersonate anyone that they want to impersonate, without authentication.

Rubric: 1 mark for idea of spoofing anyone, 1 mark for how this is done.

(e) (2 points) In general cross-origin resource sharing (CORS), i.e. allowing a script downloaded from site A to make a connection to site B, is disallowed. Explain why.

**Answer**: The risk is that site A may have malicious javacscript code, that for example will contact your bank and transfer money (provided you are logged into your bank). Preventing cross site scripting means the javascript can only access the origin site, avoiding any such risk.

Rubric: Explanation of risk with any example: 1 mark, 1 mark for how the risk is averted by blocking CORS.

(f) (2 points) Despite the above, your React application had to allow a limited form of CORS. Explain why.

**Answer**: The API is served by one application, whereas the React code is served by another application, thus they are two different sites (same machine but different port numbers also count as different sites). We therefore had to allow javascript downloaded from the React app to access the API service.

Rubric: Explanation of the problem: 1 mark, explanation of solution: 1 mark.

Note: The API site had to set a CORS header telling the browser that a fetch request from it with javascript code originating at the React server (or any server) is acceptable. This is enforced at the browser end to protect the browser user, not at the server end; the server has no way to verify what the scripts origin site was.

5. React

(a) (2 points) In the React statement `const [totalPrice, setTotalPrice] = useState(0);` explain what totalPrice and setTotalPrice are.

**Answer**: totalPrice is a state variable, while setTotalPrice() is a setter function for that state variable, which when invoked will update the value of the state variable.

Rubric: 1 mark for each part of the explanation.

(b) (1 point) In React, list items need to have a _____ attribute which is used to uniquely identify the item.

**Answer:** key

Rubric: 1 mark

(c) (2 points) In React the display is rendered from the React state. Therefore, for each *textual* form input, there should be an associated _____ which should be updated by a handler for the _____ event associated with that input.

**Answer:** state variable; onChange()

Rubric: 1 mark for each blank

(d) (6 points) Fill in the blanks for a function to update the quantity of a productId in a collection, by the amount "change".

```
const [________________, ____________________] = useState({});

const handleQuantityChange = (productId, change) => {

     setQuantities((____________________) => ({
         ...prev,
          [productId]: Math.max(0, (prev[productId] || 0) + change),
     }));
  };

  ....
 <tr key={product.product_id}>
    ......
    <span>{quantities[product.product_id] || 0}</span>

    <button onClick={() =>

    ________________________ (________________________, _________________)}>
          + </button>
 </tr>
```

**Answer:** quantities; setQuantities; prev; handleQuantityChange; product.product_id; 1

Rubric: 1 mark for each blank