Roll Number:

CS4.301: Data and Applications (Monsoon 2024)

Quiz 3

Maximum Marks: 20, Time: 45 minutes

- **Keep answers concise. State all assumptions.**
- **All questions are compulsory.**

Question 1

An online picture sharing company uses a database with the following schema:

```
create table Usr (
    uid int primary key,
    uname text not null,
    city text not null
);

create table Picture (
    pid int primary key,
    uid int not null references Usr(uid),
    size int not null,
    pdf text
);
```

Every user has a key (uid), a name (uname), and a city.

Every picture has a key (pid), an author (uid) that is a foreign key to Usr, a size, and the pdf content (which is plain text).

Consider the following:

1.1 For each query below, indicate if it is equivalent to the given query. You should answer 'yes' only if the query returns exactly the same answers (simply mention yes/no, no further elaboration is required here) (0.5 * 4 = 2 marks)

```
SELECT x.uid, x.uname,
    (SELECT count(*)
     FROM Picture y
     WHERE x.uid = y.uid AND y.size > 1000000)
FROM Usr x
WHERE x.city = 'Denver';
```

a. **Is this query equivalent?**

```
SELECT x.uid, x.uname, count(*)
FROM Usr x, Picture y
WHERE x.uid = y.uid AND y.size > 1000000 AND x.city = 'Denver'
GROUP BY x.uid, x.uname;
```

NO : 0.5
0 otherwise

b. Is this query equivalent?

```
SELECT x.uid, x.uname
FROM Usr x
WHERE x.city = 'Denver' AND exists
    (SELECT *
     FROM Picture y
     WHERE x.uid = y.uid AND y.size > 1000000);
```

NO : 0.5
0 otherwise

c. Is this query equivalent?

```
SELECT x.uid, x.uname, count(y.pid)
FROM Usr x LEFT OUTER JOIN Picture y on x.uid = y.uid AND y.size
     > 1000000
GROUP BY x.uid, x.uname, x.city
having x.city = 'Denver';
```

YES : 0.5
0 otherwise

d. Is this query equivalent?

```
SELECT x.uid, x.uname, count(*)
FROM Usr x LEFT OUTER JOIN Picture y on x.uid = y.uid AND y.size
          > 1000000
GROUP BY x.uid, x.uname, x.city
HAVING x.city = 'Denver';
```

NO : 0.5
0 otherwise

1.2 For each query below indicate if it is equivalent to the given query. You should answer 'yes' only if the query returns exactly the same answers. (simply mention yes/no, no further elaboration is required here) (0.5 * 4 = 2 marks)

```
SELECT distinct x.uid, x.uname
FROM Usr x, Picture u, Picture v, Picture w
WHERE x.uid = u.uid AND x.uid = v.uid AND x.uid = w.uid
AND u.size > 1000000 AND v.size < 3000000 AND w.size = u.size;
```

a. **Is this query equivalent?**

```
SELECT distinct x.uid, x.uname
FROM Usr x, Picture u, Picture v, Picture w
WHERE x.uid = u.uid AND x.uid = v.uid AND x.uid = w.uid
AND u.size > 1000000 AND v.size < 3000000 AND w.size > 1000000;
```

YES : 0.5
0 otherwise

b. **Is this query equivalent?**

```
SELECT distinct x.uid, x.uname
FROM Usr x, Picture u, Picture v
WHERE x.uid = u.uid AND x.uid = v.uid
AND u.size > 1000000 AND v.size < 3000000;
```

YES : 0.5
0 otherwise

c. **Is this query equivalent?**

```
SELECT distinct x.uid, x.uname
FROM Usr x, Picture u, Picture w
WHERE x.uid = u.uid AND x.uid = w.uid
AND u.size > 1000000 AND u.size < 3000000 AND u.size = w.size;
```

NO : 0.5
0 otherwise

d. **Is this query equivalent?**

```
SELECT distinct x.uid, x.uname
FROM Usr x, Picture u, Picture v, Picture w
WHERE x.uid = u.uid AND x.uid = v.uid AND x.uid = w.uid
AND u.size > 1000000 AND v.size < 3000000 AND w.size = v.size;
```

YES : 0.5
0 otherwise

Question 2 **(2)**

Consider a table People **with attributes** *SALARY* **and** *NAME*. **Given the relational algebra expression** $\Pi_{\text{NAME}}(\sigma_{\text{SALARY}>10{,}000}(\Pi_{\text{NAME}\cup\text{SALARY}}(\text{People})))$, **Which of the following are valid rewrites?** (Check all that apply.)

(simply circle the option(s) you think are correct, no further elaboration is required here)

- **a.** $\Pi_{\text{NAME}}(\sigma_{\text{SALARY}>10{,}000}(\text{People}))$
- **b.** $\Pi_{\text{NAME}}(\sigma_{\text{SALARY}>10{,}000}(\Pi_{\text{SALARY}}(\text{People})))$
- **c.** $\sigma_{\text{SALARY}>10{,}000}(\Pi_{\text{NAME}\cup\text{SALARY}}(\text{People}))$
- **d.** $\sigma_{\text{SALARY}>10{,}000}(\Pi_{\text{NAME}}(\Pi_{\text{NAME}\cup\text{SALARY}}(\text{People})))$
- **e.** $\Pi_{\text{NAME}}(\sigma_{\text{SALARY}>10{,}000}(\Pi_{\text{NAME}}(\Pi_{\text{SALARY}}(\text{People}))))$

Correct answer: (a) only

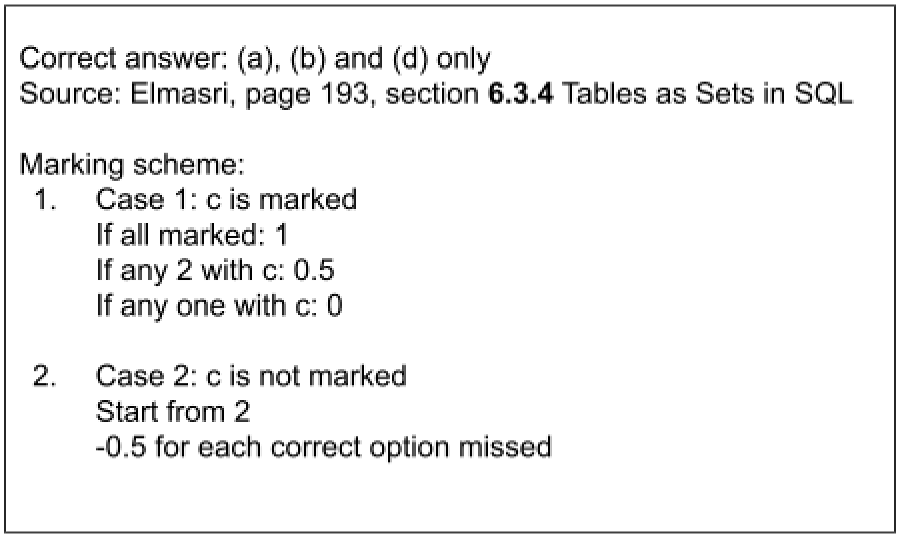
Marking scheme:
Only (a) marked: 2 marks
(a) + any one incorrect option: 1 mark
2 incorrect options marked (with or without (a)) : 0

Question 3 **(2)**

Why does SQL not automatically eliminate duplicate tuples in the results of queries? Select all the correct statement(s):

(simply circle the option(s) you think are correct, no further elaboration is required here)

- **a. Duplicate elimination is an expensive operation**
- **b. When an aggregate function is applied to tuples, in most cases we do not want to eliminate duplicates**
- **c. Duplicates are an implementation of redundancy in fault tolerance**
- **d. The user may want to see duplicate tuples in the result of a query**


Correct answer: (a), (b) and (d) only
Source: Elmasri, page 193, section **6.3.4** Tables as Sets in SQL

Marking scheme:

1. Case 1: c is marked
   If all marked: 1
   If any 2 with c: 0.5
   If any one with c: 0

2. Case 2: c is not marked
   Start from 2
   -0.5 for each correct option missed

Question 4 (4)

You are given the following database tables:

Employees

<u>EmpID</u>	EmpName	Department	Salary
1	Alice	1	50000
2	Bob	2	70000
3	Charlie	1	60000
4	David	2	80000
5	Eva	3	55000

Departments

<u>DeptID</u>	DeptName
1	HR
2	IT
3	Sales

a. **For the SQL query below, provide the CODING Order (i.e., the logical way to write the query) and the EXECUTION Order (i.e., the order in which the query processor actually executes it):**

```
SELECT Department, COUNT(*) AS EmployeeCount
FROM Employees
WHERE Salary > 60000
GROUP BY Department
HAVING COUNT(*) > 1
ORDER BY EmployeeCount DESC;
```

Correct answer:

Coding Order: The logical order to write the query:

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

Execution Order: The order in which SQL processes the query:

FROM (selects Employees table)
WHERE (filters rows with Salary > 60000)
GROUP BY (groups rows by Department)
HAVING (filters groups where COUNT(*) > 1)
SELECT (retrieves Department and EmployeeCount)
ORDER BY (orders the results by EmployeeCount in descending order)

Source: Tutorial 3
(https://courses.iiit.ac.in/mod/forum/discuss.php?d=44898#p106149)

Marking scheme: 0.5 if correct, 0 if not
for coding and execution order each

b. Examine the two SQL queries below. Identify which one(s) will fail and explain why. For each, write the expected output (if any). Keep the answer brief and concise (around 2-3 lines per query):

Query A

```
SELECT EmpName, Salary
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees)
ORDER BY Salary DESC;
```

Query B

```
SELECT DeptName, COUNT(*) AS TotalEmployees
FROM Departments
JOIN Employees ON Departments.DeptName = Employees.Department
WHERE TotalEmployees > 1
GROUP BY DeptName;
```

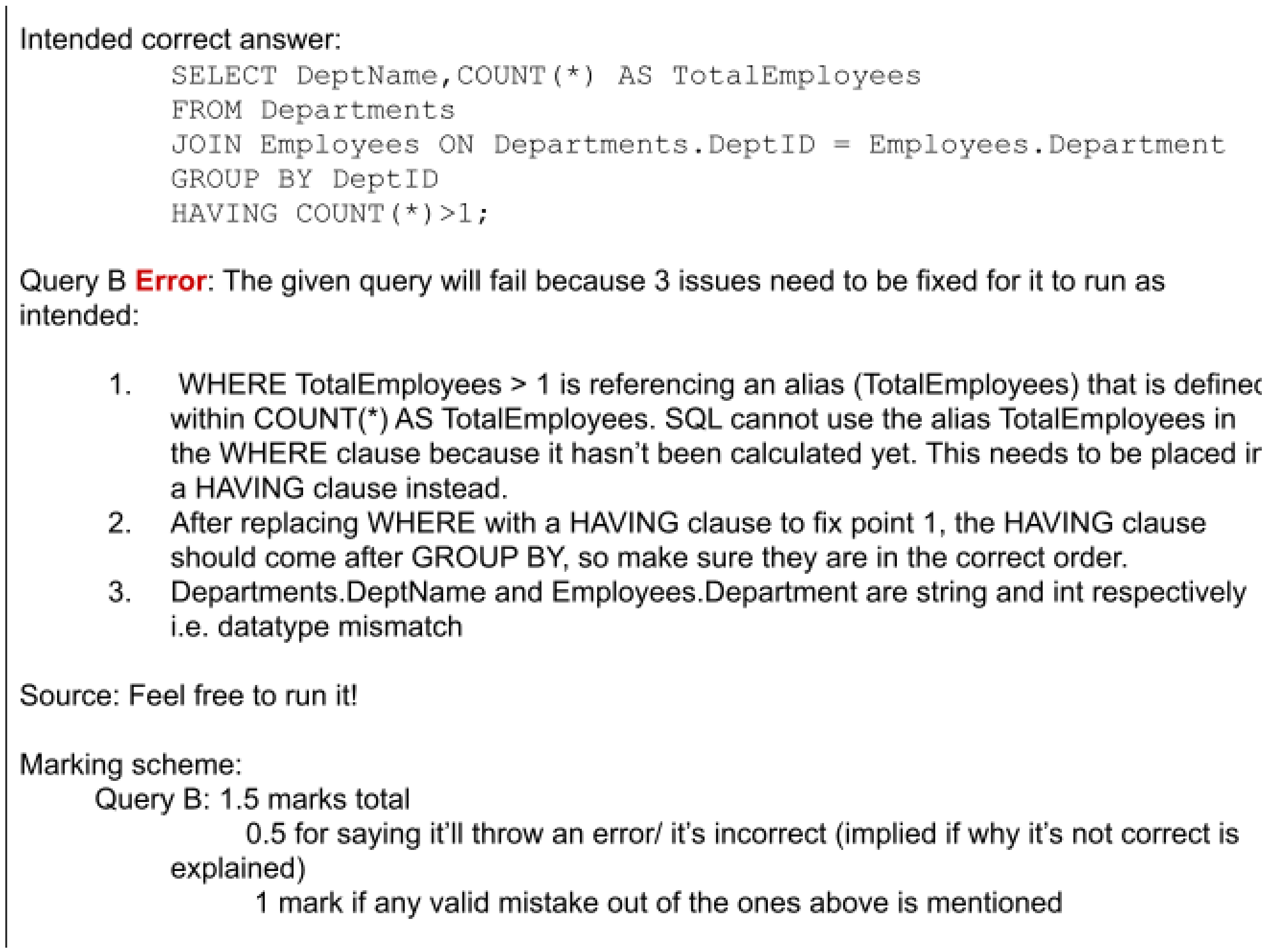
Correct answer:
Query A Output: This query is valid and returns the names and salaries of employees who earn more than the average salary in descending order. The output for the provided data would be:

```
David | 80000
Bob   | 70000
```

Source: Feel free to run the query!

Marking scheme:

- Query A: 1.5 marks total
    - 0.5 for saying it's valid (if they've provided output, even then give marks, since it's implied)
    - 1 mark for correct output (0.5 if there's only **one** error in output)


Intended correct answer:

```
        SELECT DeptName,COUNT(*) AS TotalEmployees
        FROM Departments
        JOIN Employees ON Departments.DeptID = Employees.Department
        GROUP BY DeptID
        HAVING COUNT(*)>1;
```

Query B Error: The given query will fail because 3 issues need to be fixed for it to run as intended:

1. WHERE TotalEmployees > 1 is referencing an alias (TotalEmployees) that is definec within COUNT(*) AS TotalEmployees. SQL cannot use the alias TotalEmployees in the WHERE clause because it hasn't been calculated yet. This needs to be placed ir a HAVING clause instead.
2. After replacing WHERE with a HAVING clause to fix point 1, the HAVING clause should come after GROUP BY, so make sure they are in the correct order.
3. Departments.DeptName and Employees.Department are string and int respectively i.e. datatype mismatch

Source: Feel free to run it!

Marking scheme:

Query B: 1.5 marks total

0.5 for saying it'll throw an error/ it's incorrect (implied if why it's not correct is explained)

1 mark if any valid mistake out of the ones above is mentioned

Question 5 **(4)**

You are given two matrices `A`, `B` **with schemas:**

```
A(i,j,v)

B(j,k,v)
```

Write an SQL query that computes the product matrix *`(A.B)`***. Your query should return a set of triples** `(i, k, v)` **representing the product matrix; you do not need to remove the entries with value 0. Assume 1 indexing of `i`, `j` and `k`. For example:**

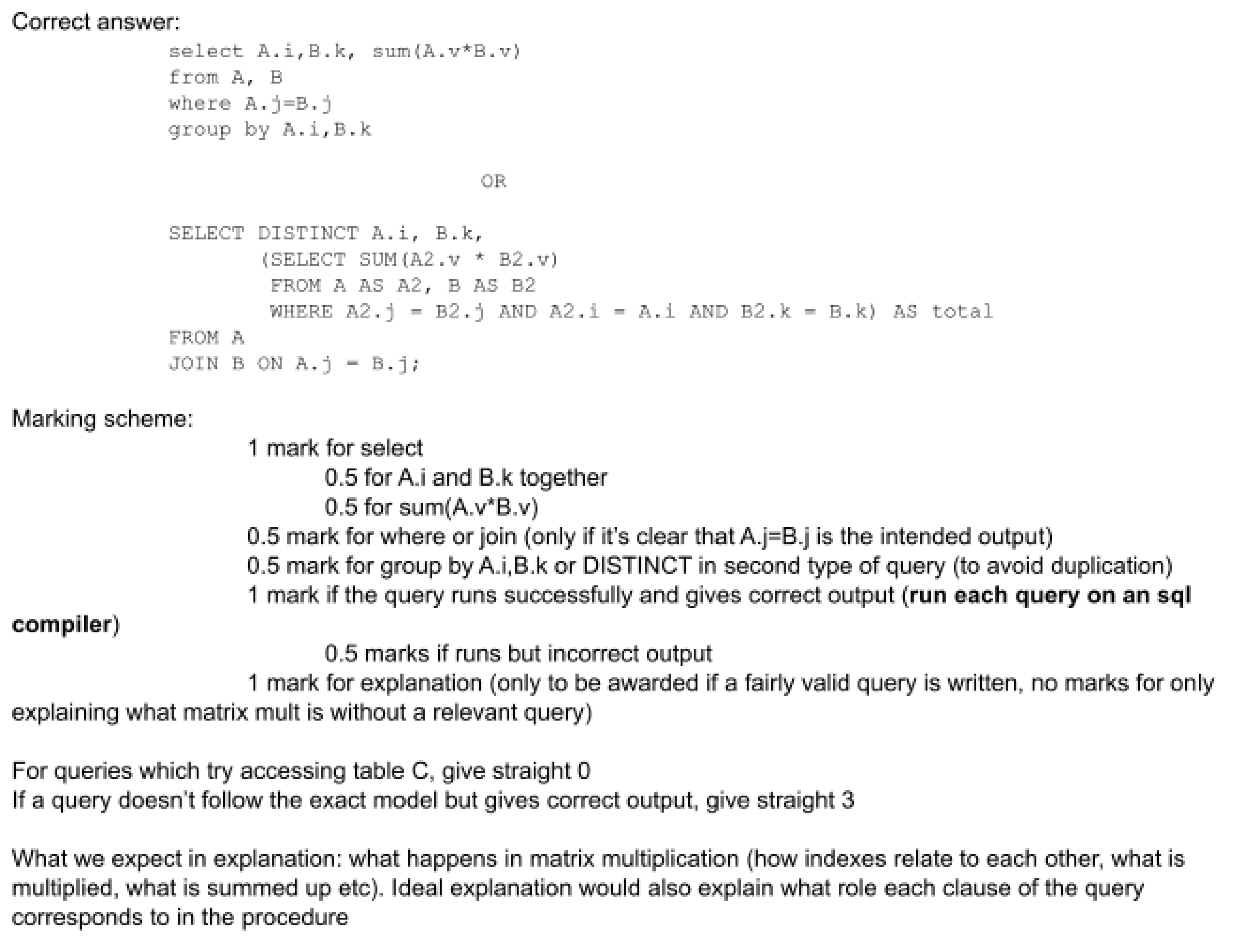
$$\begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 4 \\ -2 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 10 \\ -7 & 2 \end{pmatrix}$$

A		
i	j	v
1	1	2
1	2	1
2	1	-1
2	2	3

B		
j	k	v
1	1	1
1	2	4
2	1	-2
2	2	2

C		
i	k	v
1	1	0
1	2	10
2	1	-7
2	2	2

(Please provide concise 2-3 lines of reasoning/justification for your final answer)



Correct answer:

```
select A.i,B.k, sum(A.v*B.v)
from A, B
where A.j=B.j
group by A.i,B.k

                    OR

SELECT DISTINCT A.i, B.k,
       (SELECT SUM(A2.v * B2.v)
        FROM A AS A2, B AS B2
        WHERE A2.j = B2.j AND A2.i = A.i AND B2.k = B.k) AS total
FROM A
JOIN B ON A.j = B.j;
```

Marking scheme:

1 mark for select
- 0.5 for A.i and B.k together
- 0.5 for sum(A.v*B.v)

0.5 mark for where or join (only if it's clear that A.j=B.j is the intended output)

0.5 mark for group by A.i,B.k or DISTINCT in second type of query (to avoid duplication)

1 mark if the query runs successfully and gives correct output (**run each query on an sql compiler**)
- 0.5 marks if runs but incorrect output

1 mark for explanation (only to be awarded if a fairly valid query is written, no marks for only explaining what matrix mult is without a relevant query)

For queries which try accessing table C, give straight 0
If a query doesn't follow the exact model but gives correct output, give straight 3

What we expect in explanation: what happens in matrix multiplication (how indexes relate to each other, what is multiplied, what is summed up etc). Ideal explanation would also explain what role each clause of the query corresponds to in the procedure