### 1 Query Execution

1.1

a) **Description**
   From the *instructor* table find the name(s) of the instructor(s) whose *id* is 274.

b) **SQL (given)**
   
   ```sql
   SELECT name FROM instructor
   WHERE id=274
   ```

c) **Relational Algebra**
   
   \[ \Pi_{\text{name}} (\sigma_{id=274} \text{instructor}) \]

d) **Output**
   
<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clifton</td>
</tr>
</tbody>
</table>

1.2

*Note: There is no projection operator in the given relational algebra, so every column will be returned.*

a) **Description**
   Select rows from the *department* table where *budget* is greater than 1000000.

b) **SQL**
   
   ```sql
   SELECT * FROM department
   WHERE budget > 1000000
   ```
   
or
   ```sql
   SELECT dept_name, building, budget FROM department
   WHERE budget > 1000000
   ```

c) **Relational Algebra (given)**
   
   \[ \sigma_{\text{budget}>1000000} \text{department} \]

d) **Output**
   
   “Null” or “empty” or “∅” or “nothing is returned” or

<table>
<thead>
<tr>
<th>dept_name</th>
<th>building</th>
<th>budget</th>
</tr>
</thead>
</table>

   All of these are acceptable.
1.3

a) **Description**
   Return the sum of the budgets for all departments.

b) **SQL**
   ```sql
   SELECT sum(budget)
   FROM department;
   ```

c) **Relational Algebra** (given)
   \( \Pi_{sum(budget)} \) department

d) **Output**
<table>
<thead>
<tr>
<th>sum(budget)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2600000</td>
</tr>
</tbody>
</table>

1.4

*Note:* You will lose points if SQL query, relational algebra, and output do not match each other.

a) **Description** (given)
   Return the average salary in each department.

b) **Ideal solution**

b) **SQL**
   ```sql
   SELECT d.dept_name AS dept_name, avg(salary) AS avg_salary
   FROM department d
   LEFT JOIN instructor i
   ON d.dept_name = i.dept_name
   GROUP BY department.dept_name
   ```

c) **Relational Algebra**
   \( \Pi_{dept\_name, \ avg(salary)}(\gamma_{dept\_name, \ avg(salary)}(department \bowtie\instructor)) \)

b) **Output**
<table>
<thead>
<tr>
<th>dept_name</th>
<th>avg_salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>253433.29</td>
</tr>
<tr>
<td>Mathematics</td>
<td>135651.95</td>
</tr>
<tr>
<td>Physics</td>
<td>NULL</td>
</tr>
<tr>
<td>Statistics</td>
<td>197675.46</td>
</tr>
</tbody>
</table>
1.4.2 Acceptable solution

b) SQL

\[
\begin{align*}
\text{SELECT} & \quad \text{AVG(salary)} \\
\text{FROM} & \quad \text{instructor} \\
\text{GROUP BY} & \quad \text{dept\_name}
\end{align*}
\]

b) Relational Algebra

\[\gamma_{\text{dept\_name}, \text{avg(salary)}} \ \text{instructor}\]

b) Output

<table>
<thead>
<tr>
<th>avg(salary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>253433.29</td>
</tr>
<tr>
<td>135651.95</td>
</tr>
<tr>
<td>197675.46</td>
</tr>
</tbody>
</table>

1.5

Note: You will lose points if you forget to mention any one of the following: department name, id, budget, or salary.

a) Description

Returns the department name, id, budget, and salary for all instructors.

b) SQL

\[
\begin{align*}
\text{SELECT} & \quad \text{d.dept\_name, id, budget, salary} \\
\text{FROM} & \quad \text{department d, instructor i} \\
\text{WHERE} & \quad \text{d.dept\_name = i.dept\_name;}
\end{align*}
\]

c) Relational Algebra (given)

\[
\Pi_{\text{department.dept\_name}, \text{id}, \text{budget}, \text{salary}} (\sigma_{\text{department.dept\_name=\text{instructor.dept\_name}} \ (\text{department} \times \text{instructor}))
\]

d) Output

<table>
<thead>
<tr>
<th>dept_name</th>
<th>id</th>
<th>budget</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>274</td>
<td>800000</td>
<td>230326.16</td>
</tr>
<tr>
<td>Computer Science</td>
<td>271</td>
<td>800000</td>
<td>276520.42</td>
</tr>
<tr>
<td>Statistics</td>
<td>295</td>
<td>500000</td>
<td>277834.37</td>
</tr>
<tr>
<td>Statistics</td>
<td>307</td>
<td>500000</td>
<td>117516.56</td>
</tr>
<tr>
<td>Mathematics</td>
<td>528</td>
<td>600000</td>
<td>135651.95</td>
</tr>
</tbody>
</table>
1.6

a) **Description** (given)
   Compute the “excess” budget for each department, after subtracting the salaries for that department.

b) **SQL**

   ```sql
   SELECT department.dept_name, (budget - IFNULL(sum_salary, 0)) AS excess_budget
   FROM department
   LEFT OUTER JOIN (SELECT dept_name, SUM(salary) AS sum_salary
                    FROM instructor
                    GROUP BY dept_name) AS i
   ON department.dept_name = i.dept_name;
   ```

   *Note:* You will lose points for the sql query if your query output does not match the output provided. Since this was a tough question, you will get points for your sql query even if it does not print excess value of physics department. You will lose points in the output part of the question for missing excess budget for physics department.

c) **Relational Algebra**

   $$\gamma_{\text{department.dept\_name}, \text{budget-sum(ifnull(salary,0))}}(\sigma_{\text{department.dept\_name}=\text{instructor.dept\_name}}(\text{department } \Join \text{instructor}))$$

   *Note:* If you missed `ifnull` condition, then you lost the points for this part. If you relational algebra does not print Physics department’s excess budget, then you lost points for the question.

d) **Output**

<table>
<thead>
<tr>
<th>dept_name</th>
<th>excess_budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>293133.42</td>
</tr>
<tr>
<td>Statistics</td>
<td>104649.07</td>
</tr>
<tr>
<td>Physics</td>
<td>700000.00</td>
</tr>
<tr>
<td>Mathematics</td>
<td>464348.05</td>
</tr>
</tbody>
</table>

   *Note:* You need to print the excess budget for the physics department to get full credit for output as mentioned on piazza.

1.7

a) **Description**
   Return the *id* and the *name* of the instructor(s) with the highest salary.

b) **SQL** (given)

   ```sql
   SELECT id, name
   FROM instructor i, (SELECT max(salary) mx FROM instructor) m
   WHERE i.salary = m.mx
   ```

c) **Relational Algebra**

   $$\Pi_{\text{id, name}}(\sigma_{\text{i.salary}=\text{m.mx}}(\rho_{i}(\text{instructor}) \times \rho_{m(\text{mx})}(\Pi_{\text{max(salary)}}(\text{instructor}))))$$
   or
   $$\Pi_{\text{id, name}}(\rho_{i}(\text{instructor}) \bowtie_{i.\text{salary}=\text{m.mx}}(\rho_{m(\text{mx})}(\Pi_{\text{max(salary)}}(\text{instructor}))))$$

d) **Output**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>295</td>
<td>Cleveland</td>
</tr>
</tbody>
</table>
1.8

a) **Description** (given)
   List the highest paid instructor in each department.

b) **SQL**

   ```sql
   SELECT id, name
   FROM instructor i, (SELECT dept_name, max(salary) mx
   FROM instructor
   GROUP BY dept_name) d
   WHERE i.dept_name=d.dept_name AND i.salary=d.mx)
   ```

c) **Relational Algebra**

   $\Pi_{id,name}(\sigma_{i.salary=d.mx \land i.dept_name=d.dept_name}(\rho_i(instructor) \times \rho_d(dept_name,mx)(\gamma_{dept_name,max(salary)}instructor)))$
   or
   
   $\Pi_{id,name}(\rho_i(instructor) \bowtie_{i.salary=d.mx \land i.dept_name=d.dept_name} \rho_d(dept_name,mx)(\gamma_{dept_name,max(salary)}instructor))$

d) **Output**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>295</td>
<td>Cleveland</td>
</tr>
<tr>
<td>271</td>
<td>Li</td>
</tr>
<tr>
<td>528</td>
<td>Li</td>
</tr>
</tbody>
</table>
2 Set vs. Multiset Semantics

2.1

This question requires a DDL to enforce set semantics so the statement we are looking at would be `CREATE TABLE`. (0 points if you only mention `SELECT DISTINCT`). In SQL, duplicates are allowed except for the keys. In order to have set semantics for the entire table, either `PRIMARY KEY` or `UNIQUE` can be used in DDL and this should be applied to the full set of the attributes. (One point is deducted for not applying to the full set of attributes)

2.2

With $\Pi_{\text{dept\_name}}(\text{instructor})$, we have 3 tuples (e.g., Computer Science, Statistics, Mathematics) because we are using a set semantics (no duplicates), and the following group by will output 1 for each of them.

<table>
<thead>
<tr>
<th>Department</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>1</td>
</tr>
<tr>
<td>Statistics</td>
<td>1</td>
</tr>
<tr>
<td>Mathematics</td>
<td>1</td>
</tr>
</tbody>
</table>

2.3 Multiset Semantics

As written, the query:

```
SELECT dept_name, count(dept_name)
FROM (SELECT dept_name FROM instructor);
```

should give an error under ANSI standard SQL. Aggregates and non-aggregates can only be combined in the `SELECT` when the non-aggregates are part of a group-by or having clause.

In practice, many implementations allow this. The problem is that semantically, it says we get one `dept_name` and the count of all `dept_names`, so it really gives a row count.

<table>
<thead>
<tr>
<th>Department</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics</td>
<td>5</td>
</tr>
</tbody>
</table>

But which `dept_name`? Since SQL is unordered, there is no unambiguous “correct” answer (in practice, it is likely either the first or last tuple processed, which as we’ll see, can change depending on how you do the processing), which is why the ANSI standard says it should be an error.

Some people suggested that this would return all of the rows of the subquery, but with 5 for the count for each of them. This seems an unlikely semantics, since it requires processing all rows to get the count before we output the first row. We’ll see later why this doesn’t match the way we implement query processing, but if you can point to documentation or a reasonably widely-used implementation that follows such a semantics, we’ll regrade.

It should have been:

```
SELECT dept_name, count(dept_name)
FROM (SELECT dept_name FROM instructor)
GROUP BY dept_name;
```

<table>
<thead>
<tr>
<th>Department</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Science</td>
<td>2</td>
</tr>
<tr>
<td>Statistics</td>
<td>2</td>
</tr>
<tr>
<td>Mathematics</td>
<td>1</td>
</tr>
</tbody>
</table>
2.4 Multiset to Set

The key is that the result of the subquery under set semantics (the theoretical / relational algebra model) the subquery would return only three rows (one for each department); the easiest solution is to use DISTINCT to force this:

```sql
SELECT dept_name, count(dept_name)
FROM (SELECT DISTINCT dept_name FROM instructor)
GROUP BY dept_name;
```

Note that the only place duplicates occur in relational algebra is when we do a projection, in SQL we need to apply the DISTINCT keyword to every projection that could have a duplicate. In this case, the outer projection, since it is part of a group by, can’t return duplicates—so we can skip the distinct there. But in general, if we want set semantics throughout, the easiest solution is to replace every SELECT with SELECT DISTINCT.

While it is possible to get a semantically equivalent query for this query by placing the DISTINCT in the count instead—`count(DISTINCT dept_name)`—this is a less general approach, as it only works if we have aggregation in appropriate places.