DESCRIPTION

Table: Employees

Column Name	Type
employee_id	int
name	varchar
reports_to	int
age	int

employee_id is the column with unique values for this table.

This table contains information about the employees and the id of the manager they report to. Some employees do not report to anyone (reports_to is null).

For this problem, we will consider a **manager** an employee who has at least 1 other employee reporting to them.

Write a solution to report the ids and the names of all **managers**, the number of employees who report **directly** to them, and the average age of the reports rounded to the nearest integer.

Return the result table ordered by employee_id.

The result format is in the following example.

Example 1:

Input:

Employees table:

employee_id	name	reports_to	age
9	Hercy	null	43
6	Alice	9	41
4	Bob	9	36

```
| 2           | Winston | null       | 37  |
+-------------+---------+------------+-----+
```

Output:

```
+-------------+-------+---------------+-------------+
| employee_id | name  | reports_count | average_age |
+-------------+-------+---------------+-------------+
| 9           | Hercy | 2             | 39          |
+-------------+-------+---------------+-------------+
```

Explanation: Hercy has 2 people report directly to him, Alice and Bob. Their average age is (41+36)/2 = 38.5, which is 39 after rounding it to the nearest integer.

Example 2:

Input:

Employees table:

```
+-------------+---------+------------+-----+
| employee_id | name    | reports_to | age |
|-------------|---------|------------|-----|
| 1           | Michael | null       | 45  |
| 2           | Alice   | 1          | 38  |
| 3           | Bob     | 1          | 42  |
| 4           | Charlie | 2          | 34  |
| 5           | David   | 2          | 40  |
| 6           | Eve     | 3          | 37  |
| 7           | Frank   | null       | 50  |
| 8           | Grace   | null       | 48  |
+-------------+---------+------------+-----+
```

Output:

```
+-------------+---------+---------------+-------------+
| employee_id | name    | reports_count | average_age |
| ----------- | ------- | ------------- | ----------- |
| 1       | Michael | 2         | 40      |
| 2       | Alice   | 2         | 37      |
| 3       | Bob     | 1         | 37      |
+-------------+---------+---------------+-------------+
```

SOLUTION

MySQL:

- Self join Employees table as e1 and e2 where `e1.reports_to` equals `e2.employee_id`
- Select the desired columns from the joined table in which COUNT is used to get reports_counts, AVG is used to get average_age and ROUND for the nearest integer
- Group by employee_id and order by employee_id

```
SELECt e1.reports_to employee_id, e2.name, COUNT(e1.employee_id) reports_count,
ROUND(AVG(e1.age), 0) average_age
FROM Employees e1
JOIN Employees e2
ON e1.reports_to = e2.employee_id
GROUP BY employee_id
ORDER BY employee_id;
```

PostgreSQL:

- Similar approach as above except a left join is used for joining Employees tables
- Must add a WHERE clause where e1.reports to is not null

```
SELECT e1.reports_to employee_id, e2.name, COUNT(e1.reports_to) reports_count,
ROUND(AVG(e1.age)) average_age
FROM Employees e1
LEFT JOIN Employees e2
ON e1.reports_to = e2.employee_id
WHERE e1.reports_to IS NOT NULL
GROUP BY 1, 2
ORDER BY 1;
```