120 minutes

SQL Coding Quiz

* Required

* This form will record your name, please fill your name.

Average Salary

1

Given two tables below, write a query to display the comparison result (higher/lower/same) of the average salary of employees in a department to the company's average salary.

* (5 Points)

```
-- Table: salary
-- | id | employee_id | amount | pay_date   |
-- |----|-------------|--------|------------|
-- | 1  | 1           | 9000   | 2017-03-31 |
-- | 2  | 2           | 6000   | 2017-03-31 |
-- | 3  | 3           | 10000  | 2017-03-31 |
-- | 4  | 1           | 7000   | 2017-02-28 |
-- | 5  | 2           | 6000   | 2017-02-28 |
-- | 6  | 3           | 8000   | 2017-02-28 |


-- The employee_id column refers to the employee_id in the following table employee.


-- | employee_id | department_id |
-- |-------------|---------------|
-- | 1           | 1             |
-- | 2           | 2             |
-- | 3           | 2             |


-- So for the sample data above, the result is:


-- | pay_month | department_id | comparison  |
-- |-----------|---------------|-------------|
-- | 2017-03   | 1             | higher      |
-- | 2017-03   | 2             | lower       |
-- | 2017-02   | 1             | same        |
-- | 2017-02   | 2             | same        |
```

Quite Student

2

Write an SQL query to report the students (student_id, student_name) being "quiet" in ALL exams. A "quite" student is the one who took at least one exam and didn't score neither the high score nor the low score. *

(5 Points)

```
-- A "quite" student is the one who took at least one exam and didn't score
neither the high score nor the low score.

-- Write an SQL query to report the students (student_id, student_name)
being "quiet" in ALL exams.

-- Don't return the student who has never taken any exam. Return the result
table ordered by student_id.

-- The query result format is in the following example.

-- Student table:
-- +-------------+---------------+
-- | student_id  | student_name  |
-- +-------------+---------------+
-- | 1           | Daniel        |
-- | 2           | Jade          |
-- | 3           | Stella        |
-- | 4           | Jonathan      |
-- | 5           | Will          |
-- +-------------+---------------+

-- Exam table:
-- +------------+--------------+-----------+
-- | exam_id    | student_id   | score     |
-- +------------+--------------+-----------+
-- | 10         |     1        |    70     |
-- | 10         |     2        |    80     |
-- | 10         |     3        |    90     |
-- | 20         |     1        |    80     |
-- | 30         |     1        |    70     |
-- | 30         |     3        |    80     |
-- | 30         |     4        |    90     |
-- | 40         |     1        |    60     |
-- | 40         |     2        |    70     |
-- | 40         |     4        |    80     |
-- +------------+--------------+-----------+

-- Result table:
-- +-------------+---------------+
-- | student_id  | student_name  |
-- +-------------+---------------+
-- | 2           | Jade          |
-- +-------------+---------------+
```

Consecutive Rows

3

Write a query to display the records which have 3 or more consecutive rows and the amount of people more than 100(inclusive). *
(5 Points)

```
-- X city built a new stadium, each day many people visit it and
the stats are saved as these columns: id, visit_date, people

-- Please write a query to display the records which have 3 or
more consecutive rows and the amount of people more than 100(inclusive).

-- For example, the table stadium:
-- +------+------------+-----------+
-- | id   | visit_date | people    |
-- +------+------------+-----------+
-- | 1    | 2017-01-01 | 10        |
-- | 2    | 2017-01-02 | 109       |
-- | 3    | 2017-01-03 | 150       |
-- | 4    | 2017-01-04 | 99        |
-- | 5    | 2017-01-05 | 145       |
-- | 6    | 2017-01-06 | 1455      |
-- | 7    | 2017-01-07 | 199       |
-- | 8    | 2017-01-08 | 188       |
-- +------+------------+-----------+
-- For the sample data above, the output is:

-- +------+------------+-----------+
-- | id   | visit_date | people    |
-- +------+------------+-----------+
-- | 5    | 2017-01-05 | 145       |
-- | 6    | 2017-01-06 | 1455      |
-- | 7    | 2017-01-07 | 199       |
-- | 8    | 2017-01-08 | 188       |
-- +------+------------+-----------+
-- Note:
-- Each day only have one row record, and the dates are increasing with id increasing.
```

Guess Who?

4

Write an SQL query to find how many users visited the bank and didn't do any transactions, how many visited the bank and did one transaction and so on. *
(5 Points)

```
-- Write an SQL query to find how many users visited the bank and didn't do any transactions,
how many visited the bank and did one transaction and so on.

-- The query result format is in the following example:

-- Visits table:
-- +---------+------------+
-- | user_id | visit_date |
-- +---------+------------+
-- | 1       | 2020-01-01 |
-- | 2       | 2020-01-02 |
-- | 12      | 2020-01-01 |
-- | 19      | 2020-01-03 |
-- | 1       | 2020-01-02 |
-- | 2       | 2020-01-03 |
-- | 1       | 2020-01-04 |
-- | 7       | 2020-01-11 |
-- | 9       | 2020-01-25 |
-- | 8       | 2020-01-28 |
-- +---------+------------+
-- Transactions table:
-- +---------+------------------+--------+
-- | user_id | transaction_date | amount |
-- +---------+------------------+--------+
-- | 1       | 2020-01-02       | 120    |
-- | 2       | 2020-01-03       | 22     |
-- | 7       | 2020-01-11       | 232    |
-- | 1       | 2020-01-04       | 7      |
-- | 9       | 2020-01-25       | 33     |
-- | 9       | 2020-01-25       | 66     |
-- | 8       | 2020-01-28       | 1      |
-- | 9       | 2020-01-25       | 99     |
-- +---------+------------------+--------+
-- Result table:
-- +--------------------+--------------+
-- | transactions_count | visits_count |
-- +--------------------+--------------+
-- | 0                  | 4            |
-- | 1                  | 5            |
-- | 2                  | 0            |
-- | 3                  | 1            |
-- +--------------------+--------------+
-- * For transactions_count = 0, The visits (1, "2020-01-01"), (2, "2020-01-02"),
(12, "2020-01-01") and (19, "2020-01-03") did no transactions so visits_count = 4.
-- * For transactions_count = 1, The visits (2, "2020-01-03"), (7, "2020-01-11"),
(8, "2020-01-28"), (1, "2020-01-02") and (1, "2020-01-04") did one transaction so visits_count = 5.
-- * For transactions_count = 2, No customers visited the bank and
did two transactions so visits_count = 0.
-- * For transactions_count = 3, The visit (9, "2020-01-25")
did three transactions so visits_count = 1.
-- * For transactions_count >= 4, No customers visited the bank and
did more than three transactions so we will stop at transactions_count = 3
```

Generate Report

5

Write an SQL query to generate a report of period state for each continuous interval of days in the period from 2019–01–01 to 2019–12–31. * (5 Points)

```
-- A system is running one task every day. Every task is independent of the previous tasks.
The tasks can fail or succeed.

-- Write an SQL query to generate a report of period_state for each continuous interval of
days in the period from 2019-01-01 to 2019-12-31.

-- period_state is 'failed' if tasks in this interval failed or 'succeeded' if tasks
in this interval succeeded. Interval of days are retrieved as start_date and end_date.

-- Order result by start_date.

-- The query result format is in the following example:

-- Failed table:
-- +-------------------+
-- | fail_date         |
-- +-------------------+
-- | 2018-12-28        |
-- | 2018-12-29        |
-- | 2019-01-04        |
-- | 2019-01-05        |
-- +-------------------+

-- Succeeded table:
-- +-------------------+
-- | success_date      |
-- +-------------------+
-- | 2018-12-30        |
-- | 2018-12-31        |
-- | 2019-01-01        |
-- | 2019-01-02        |
-- | 2019-01-03        |
-- | 2019-01-06        |
-- +-------------------+


-- Result table:
-- +--------------+--------------+--------------+
-- | period_state | start_date   | end_date     |
-- +--------------+--------------+--------------+
-- | succeeded    | 2019-01-01   | 2019-01-03   |
-- | failed       | 2019-01-04   | 2019-01-05   |
-- | succeeded    | 2019-01-06   | 2019-01-06   |
-- +--------------+--------------+--------------+

-- The report ignored the system state in 2018 as we care about the system in the
period 2019-01-01 to 2019-12-31.
-- From 2019-01-01 to 2019-01-03 all tasks succeeded and the system state was "succeeded".
-- From 2019-01-04 to 2019-01-05 all tasks failed and system state was "failed".
-- From 2019-01-06 to 2019-01-06 all tasks succeeded and system state was "succeeded".
```

Weekly Report

6

Write an SQL query to report how many units in each category have been ordered on each day of the week.

* (5 Points)

```
-- You are the business owner and would like to obtain a sales report
for category items and day of the week.

-- Write an SQL query to report how many units in each category
have been ordered on each day of the week.

-- Return the result table ordered by category.

-- The query result format is in the following example:

-- Orders table:
-- +------------+-------------+-------------+-------------+-------------+
-- | order_id   | customer_id | order_date  | item_id     | quantity    |
-- +------------+-------------+-------------+-------------+-------------+
-- | 1          | 1           | 2020-06-01  | 1           | 10          |
-- | 2          | 1           | 2020-06-08  | 2           | 10          |
-- | 3          | 2           | 2020-06-02  | 1           | 5           |
-- | 4          | 3           | 2020-06-03  | 3           | 5           |
-- | 5          | 4           | 2020-06-04  | 4           | 1           |
-- | 6          | 4           | 2020-06-05  | 5           | 5           |
-- | 7          | 5           | 2020-06-05  | 1           | 10          |
-- | 8          | 5           | 2020-06-14  | 4           | 5           |
-- | 9          | 5           | 2020-06-21  | 3           | 5           |
-- +------------+-------------+-------------+-------------+-------------+

-- Items table:
-- +------------+--------------------+---------------+
-- | item_id    | item_name          | item_category |
-- +------------+--------------------+---------------+
-- | 1          | LC Alg. Book       | Book          |
-- | 2          | LC DB. Book        | Book          |
-- | 3          | LC SmarthPhone     | Phone         |
-- | 4          | LC Phone 2020      | Phone         |
-- | 5          | LC SmartGlass      | Glasses       |
-- | 6          | LC T-Shirt XL      | T-Shirt       |
-- +------------+--------------------+---------------+

-- Result table:
-- +------------+--------+---------+-----------+----------+--------+----------+--------+
-- | Category   | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
-- +------------+--------+---------+-----------+----------+--------+----------+--------+
-- | Book       | 20     | 5       | 0         | 0        | 10     | 0        | 0      |
-- | Glasses    | 0      | 0       | 0         | 0        | 5      | 0        | 0      |
-- | Phone      | 0      | 0       | 5         | 1        | 0      | 0        | 10     |
-- | T-Shirt    | 0      | 0       | 0         | 0        | 0      | 0        | 0      |
-- +------------+--------+---------+-----------+----------+--------+----------+--------+
-- On Monday (2020-06-01, 2020-06-08) were sold a total of 20 units (10 + 10)
in the category Book (ids: 1, 2).
-- On Tuesday (2020-06-02) were sold a total of 5 units
in the category Book (ids: 1, 2).
-- On Wednesday (2020-06-03) were sold a total of 5 units
in the category Phone (ids: 3, 4).
-- On Thursday (2020-06-04) were sold a total of 1 unit
in the category Phone (ids: 3, 4).
-- On Friday (2020-06-05) were sold 10 units in the category
Book (ids: 1, 2) and 5 units in Glasses (ids: 5).
-- On Saturday there are no items sold.
-- On Sunday (2020-06-14, 2020-06-21) were sold a total of
10 units (5 +5) in the category Phone (ids: 3, 4).
-- There are no sales of T-Shirt.
```

Ranking

7

Write an SQL query to find employees who earn the top three salaries in each of the department. For the above tables, your SQL query should return the following rows (order of rows does not matter). *
(5 Points)

```
-- The Employee table holds all employees. Every employee has an Id, and
there is also a column for the department Id.

-- +----+-------+--------+--------------+
-- | Id | Name  | Salary | DepartmentId |
-- +----+-------+--------+--------------+
-- | 1  | Joe   | 85000  | 1            |
-- | 2  | Henry | 80000  | 2            |
-- | 3  | Sam   | 60000  | 2            |
-- | 4  | Max   | 90000  | 1            |
-- | 5  | Janet | 69000  | 1            |
-- | 6  | Randy | 85000  | 1            |
-- | 7  | Will  | 70000  | 1            |
-- +----+-------+--------+--------------+
-- The Department table holds all departments of the company.

-- +----+----------+
-- | Id | Name     |
-- +----+----------+
-- | 1  | IT       |
-- | 2  | Sales    |
-- +----+----------+
-- Write a SQL query to find employees who earn the top three salaries in
each of the department. For the above tables, your SQL query should
return the following rows (order of rows does not matter).

-- +------------+----------+--------+
-- | Department | Employee | Salary |
-- +------------+----------+--------+
-- | IT         | Max      | 90000  |
```