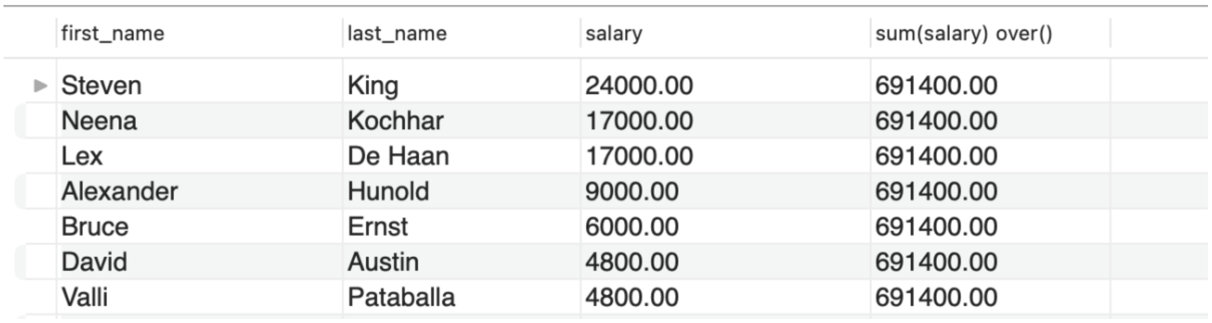
Window Function Practice Questions:

Qn 1: For each employee, find their first name, last name, salary and the sum of all salaries in the company.

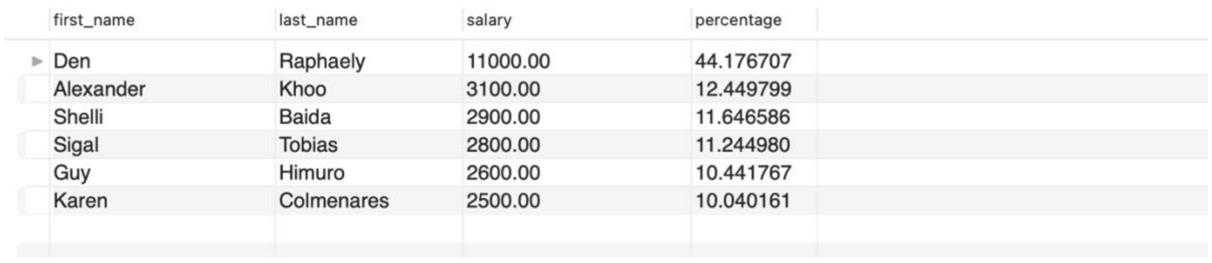
Expected output:



first_name	last_name	salary	sum(salary) over()
Steven	King	24000.00	691400.00
Neena	Kochhar	17000.00	691400.00
Lex	De Haan	17000.00	691400.00
Alexander	Hunold	9000.00	691400.00
Bruce	Ernst	6000.00	691400.00
David	Austin	4800.00	691400.00
Valli	Pataballa	4800.00	691400.00

Qn 2: (CALCULATING PERCENTAGES) For all employees from department with department_id = 30, show their first_name, last_name, salary, the % of their salary to the SUM of all salaries in that department as percentage?

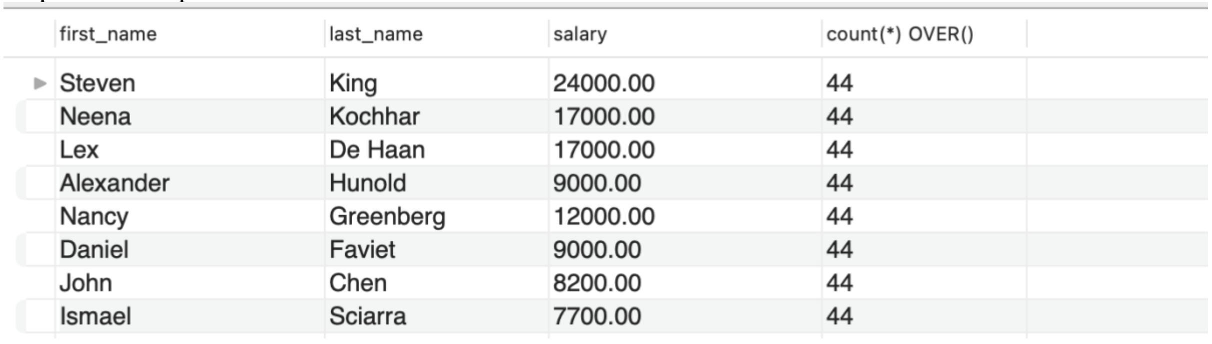
Expected output:



first_name	last_name	salary	percentage
Den	Raphaely	11000.00	44.176707
Alexander	Khoo	3100.00	12.449799
Shelli	Baida	2900.00	11.646586
Sigal	Tobias	2800.00	11.244980
Guy	Himuro	2600.00	10.441767
Karen	Colmenares	2500.00	10.040161

Qn 3: For each employee that earns more than 7000, show their first_name, last_name, salary and the number of all employees who earn more than 7000?

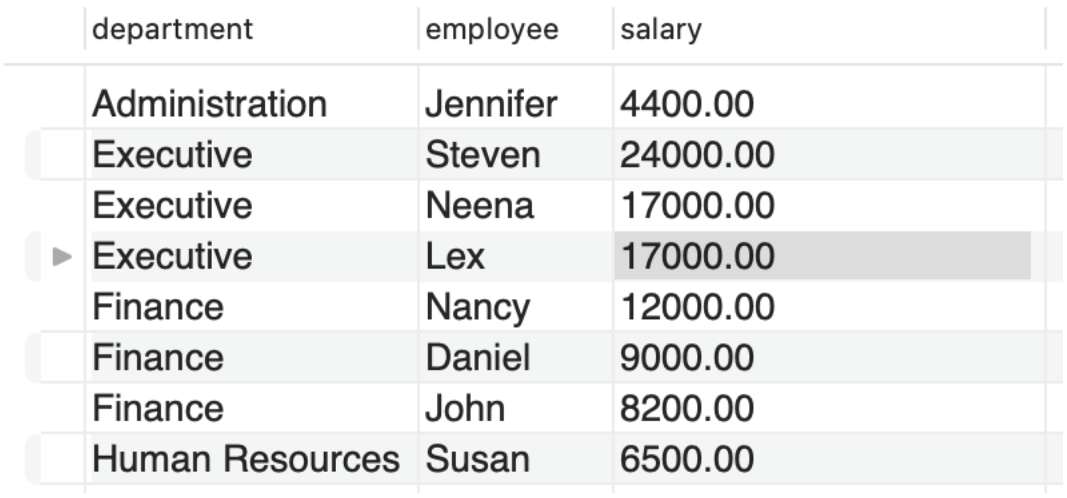
Expected Output



first_name	last_name	salary	count(*) OVER()
Steven	King	24000.00	44
Neena	Kochhar	17000.00	44
Lex	De Haan	17000.00	44
Alexander	Hunold	9000.00	44
Nancy	Greenberg	12000.00	44
Daniel	Faviet	9000.00	44
John	Chen	8200.00	44
Ismael	Sciarra	7700.00	44

Qn 4 : Write a SQL query to find employees who earn the top three salaries in each of the department.

Expected Output:



department	employee	salary
Administration	Jennifer	4400.00
Executive	Steven	24000.00
Executive	Neena	17000.00
Executive	Lex	17000.00
Finance	Nancy	12000.00
Finance	Daniel	9000.00
Finance	John	8200.00
Human Resources	Susan	6500.00

Qn 5 :

-- Write a SQL query to rank scores.

-- If there is a tie between two scores, both should have the same ranking.

-- Note that after a tie, the next ranking number should be the next consecutive integer value.

-- In other words, there should be no "holes" between ranks.

```
-- +----+-------+

-- | Id | Score |

-- +----+-------+

-- | 1  | 3.50  |

-- | 2  | 3.65  |

-- | 3  | 4.00  |

-- | 4  | 3.85  |

-- | 5  | 4.00  |

-- | 6  | 3.65  |
-- +----+-------+
```

-- For example, given the above Scores table, your query should generate the following report (order by highest score):

```
-- +-------+---------+

-- | score | Rank    |

-- +-------+---------+

-- | 4.00  | 1       |

-- | 4.00  | 1       |

-- | 3.85  | 2       |

-- | 3.65  | 3       |

-- | 3.65  | 3       |

-- | 3.50  | 4       |

-- +-------+---------+
```

-- Important Note: For MySQL solutions, to escape reserved words used as column names,

-- you can use an apostrophe before and after the keyword. For example `Rank`.

Qn 6: Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id. The output must be sorted by increasing student_id.

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

-- Table: Enrollments

```
-- +---------------+---------+

-- | Column Name   | Type    |

-- +---------------+---------+

-- | student_id    | int     |

-- | course_id     | int     |

-- | grade         | int     |
-- +---------------+---------+
-- (student_id, course_id) is the primary key of this table.
-- Enrollments table:
-- +------------+-------------------+
-- | student_id | course_id | grade |
-- +------------+-----------+-------+
-- | 2          | 2         | 95    |
-- | 2          | 3         | 95    |
-- | 1          | 1         | 90    |
-- | 1          | 2         | 99    |
-- | 3          | 1         | 80    |
-- | 3          | 2         | 75    |
-- | 3          | 3         | 82    |
-- +------------+-----------+-------+

-- Result table:
-- +------------+-------------------+
-- | student_id | course_id | grade |
-- +------------+-----------+-------+
-- | 1          | 2         | 99    |
-- | 2          | 2         | 95    |
-- | 3          | 3         | 82    |
-- +------------+-----------+-------+
```

Qn 6 : Write an SQL query to show the second most recent activity of each user.

```sql
-- If the user only has one activity, return that one.


-- A user can't perform more than one activity at the same time. Return the result table in any order.


-- Table: UserActivity


-- +---------------+---------+

-- | Column Name   | Type    |

-- +---------------+---------+

-- | username      | varchar |

-- | activity      | varchar |

-- | startDate     | Date    |

-- | endDate       | Date    |

-- +---------------+---------+

-- This table does not contain primary key.

-- This table contain information about the activity performed of each user in a period of time.

-- A person with username performed a activity from startDate to endDate.



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


-- UserActivity table:

-- +------------+--------------+-------------+-------------+

-- | username   | activity     | startDate   | endDate     |
-- +------------+--------------+-------------+-------------+

-- | Alice      | Travel       | 2020-02-12  | 2020-02-20  |

-- | Alice      | Dancing      | 2020-02-21  | 2020-02-23  |

-- | Alice      | Travel       | 2020-02-24  | 2020-02-28  |

-- | Bob        | Travel       | 2020-02-11  | 2020-02-18  |

-- +------------+--------------+-------------+-------------+


-- Result table:

-- +------------+--------------+-------------+-------------+

-- | username   | activity     | startDate   | endDate     |

-- +------------+--------------+-------------+-------------+

-- | Alice      | Dancing      | 2020-02-21  | 2020-02-23  |

-- | Bob        | Travel       | 2020-02-11  | 2020-02-18  |

-- +------------+--------------+-------------+-------------+
```

-- The most recent activity of Alice is Travel from 2020-02-24 to 2020-02-28, before that she was dancing from 2020-02-21 to 2020-02-23.

-- Bob only has one record, we just take that one.

Qn 7 : Write a SQL query to find all numbers that appear at least three times consecutively.

```
-- +----+-----+

-- | Id | Num |

-- +----+-----+

-- | 1  |  1  |

-- | 2  |  1  |

-- | 3  |  1  |
-- | 4  |  2  |

-- | 5  |  1  |

-- | 6  |  2  |

-- | 7  |  2  |

-- +----+-----+
```

-- For example, given the above Logs table, 1 is the only number that appears consecutively for at least three times.

```
-- +-----------------+

-- | ConsecutiveNums |

-- +-----------------+

-- | 1               |

-- +-----------------+
```

Try to use lag and lead functions to solve this.

Qn 8: Write an SQL query that selects the product id, year, quantity, and price for the first year of every product sold.

```
--
Table:
Sales

-- +-------------+-------+
-- | Column Name | Type  |
-- +-------------+-------+
-- | sale_id     | int   |
-- | product_id  | int   |
-- | year        | int   |
-- | quantity    | int   |
-- | price       | int   |
-- +-------------+-------+
-- sale_id is the primary key of this table.
-- product_id is a foreign key to Product table.
-- Note that the price is per unit.
-- Table: Product

-- +--------------+---------+
-- | Column Name  | Type    |
-- +--------------+---------+
-- | product_id   | int     |
-- | product_name | varchar |
-- +--------------+---------+
-- product_id is the primary key of this table.



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

-- Sales table:
-- +---------+------------+------+----------+-------+
-- | sale_id | product_id | year | quantity | price |
-- +---------+------------+------+----------+-------+
-- | 1       | 100        | 2008 | 10       | 5000  |
-- | 2       | 100        | 2009 | 12       | 5000  |
-- | 7       | 200        | 2011 | 15       | 9000  |
-- +---------+------------+------+----------+-------+

-- Product table:
-- +------------+--------------+
-- | product_id | product_name |
-- +------------+--------------+
-- | 100        | Nokia        |
-- | 200        | Apple        |
-- | 300        | Samsung      |
-- +------------+--------------+

-- Result table:
-- +------------+------------+----------+-------+
-- | product_id | first_year | quantity | price |
-- +------------+------------+----------+-------+
-- | 100        | 2008       | 10       | 5000  |
-- | 200        | 2011       | 15       | 9000  |
-- +------------+------------+----------+-------+
```

Qn 9 :

-- Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.

--
Table:
Product

```
-- +--------------+---------+
-- | Column Name  | Type    |
-- +--------------+---------+
-- | product_id   | int     |
-- | product_name | varchar |
-- | unit_price   | int     |
-- +--------------+---------+
-- product_id is the primary key of this table.
-- Table: Sales

-- +-------------+---------+
-- | Column Name | Type    |
-- +-------------+---------+
-- | seller_id   | int     |
-- | product_id  | int     |
-- | buyer_id    | int     |
-- | sale_date   | date    |
-- | quantity    | int     |
-- | price       | int     |
-- +------ ------+---------+
-- This table has no primary key, it can have repeated rows.
-- product_id is a foreign key to Product table.



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

-- Product table:
-- +------------+--------------+------------+
-- | product_id | product_name | unit_price |
-- +------------+--------------+------------+
-- | 1          | S8           | 1000       |
-- | 2          | G4           | 800        |
-- | 3          | iPhone       | 1400       |
-- +------------+--------------+------------+
-- Sales table:
-- +-----------+------------+----------+------------+----------+-------+
-- | seller_id | product_id | buyer_id | sale_date  | quantity | price |
-- +-----------+------------+----------+------------+----------+-------+
-- | 1         | 1          | 1        | 2019-01-21 | 2        | 2000  |
-- | 1         | 2          | 2        | 2019-02-17 | 1        | 800   |
-- | 2         | 2          | 3        | 2019-06-02 | 1        | 800   |
-- | 3         | 3          | 4        | 2019-05-13 | 2        | 2800  |
-- +-----------+------------+----------+------------+----------+-------+

-- Result table:
-- +-------------+
-- | seller_id   |
-- +-------------+
-- | 1           |
-- | 3           |
-- +-------------+
-- Both sellers with id 1 and 3 sold products with the most total price of 2800.
```

Qn 10: Write an SQL query that reports all the projects that have the most employees.

Table:
Project

```
-- +-------------+---------+
-- | Column Name | Type    |
-- +-------------+---------+
-- | project_id  | int     |
-- | employee_id | int     |
-- +-------------+---------+
-- (project_id, employee_id) is the primary key of this table.
-- employee_id is a foreign key to Employee table.
-- Table: Employee


-- +------------------+---------+
-- | Column Name      | Type    |
-- +------------------+---------+
-- | employee_id      | int     |
-- | name             | varchar |
-- | experience_years | int     |
-- +------------------+---------+
-- employee_id is the primary key of this table.



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

-- Project table:
```

```
-- +-------------+-------------+
-- | project_id  | employee_id |
-- +-------------+-------------+
-- | 1           | 1           |
-- | 1           | 2           |
-- | 1           | 3           |
-- | 2           | 1           |
-- | 2           | 4           |
-- +-------------+-------------+

-- Employee table:
-- +-------------+--------+------------------+
-- | employee_id | name   | experience_years |
-- +-------------+--------+------------------+
-- | 1           | Khaled | 3                |
-- | 2           | Ali    | 2                |
-- | 3           | John   | 1                |
-- | 4           | Doe    | 2                |
-- +-------------+--------+------------------+

-- Result table:
-- +-------------+
-- | project_id  |
-- +-------------+
-- | 1           |
-- +-------------+
-- The first project has 3 employees while the second one has 2.
```