###sql exercise

USE employees;

# Exercise 1

# Find the average salary of male and female employees in each department.

SELECT

d.dept\_name, e.gender, AVG(salary)

FROM

salaries s

JOIN

employees e ON s.emp\_no = e.emp\_no

JOIN

dept\_emp de ON e.emp\_no = de.emp\_no

JOIN

departments d ON d.dept\_no = de.dept\_no

GROUP BY de.dept\_no , e.gender

ORDER BY de.dept\_no;

# Exercise 2

# Find the lowest department number encountered the 'dept\_emp' table. Then, find the highest department number.

SELECT

MIN(dept\_no)

FROM

dept\_emp;

SELECT

MAX(dept\_no)

FROM

dept\_emp;

# Exercise 3

# Obtain a table containing the following three fields for all individuals whose employee number is no greater than 10040:

# - employee number

# - the smallest department number among the departments where an employee has worked in (use a subquery to retrieve this value from the 'dept\_emp' table)

# - assign '110022' as 'manager' to all individuals whose employee number is less than or equal to 10020, and '110039' to those whose number is between 10021 and 10040 inclusive (use a CASE statement to create the third field).

# If you've worked correctly, you should obtain an output containing 40 rows.

# Here’s the top part of the output.

SELECT

emp\_no,

(SELECT

MIN(dept\_no)

FROM

dept\_emp de

WHERE

e.emp\_no = de.emp\_no) dept\_no,

CASE

WHEN emp\_no <= 10020 THEN '110022'

ELSE '110039'

END AS manager

FROM

employees e

WHERE

emp\_no <= 10040;

# Exercise 4

# Retrieve a list with all employees that have been hired in the year 2000.

SELECT

\*

FROM

employees

WHERE

YEAR(hire\_date) = 2000;

# Exercise 5

# Retrieve a list with all employees from the ‘titles’ table who are engineers.

# Repeat the exercise, this time retrieving a list with all employees from the ‘titles’ table who are senior engineers.

SELECT

\*

FROM

titles

WHERE

title LIKE ('%engineer%');

SELECT

\*

FROM

titles

WHERE

title LIKE ('%senior engineer%');

# After LIKE, you could proceed to indicate what you are looking for with or without using parentheses. Both options work correctly. We think using parentheses is better for legibility reasons and that’s why it is the first option we’ve suggested.

SELECT

\*

FROM

titles

WHERE

title LIKE '%engineer%';

SELECT

\*

FROM

titles

WHERE

title LIKE '%senior engineer%';

# Exercise 6

# Create a procedure that asks you to insert an employee number to obtain an output containing the same number, as well as the number and name of the last department the employee has worked for.

# Finally, call the procedure for employee number 10010.

# If you've worked correctly, you should see that employee number 10010 has worked for department number 6 - "Quality Management".

DROP procedure IF EXISTS last\_dept;

DELIMITER $$

CREATE PROCEDURE last\_dept (in p\_emp\_no integer)

BEGIN

SELECT

e.emp\_no, d.dept\_no, d.dept\_name

FROM

employees e

JOIN

dept\_emp de ON e.emp\_no = de.emp\_no

JOIN

departments d ON de.dept\_no = d.dept\_no

WHERE

e.emp\_no = p\_emp\_no

AND de.from\_date = (SELECT

MAX(from\_date)

FROM

dept\_emp

WHERE

emp\_no = p\_emp\_no);

END$$

DELIMITER ;

call employees.last\_dept(10010);

# Exercise 7

# How many contracts have been registered in the ‘salaries’ table with duration of more than one year and of value higher than or equal to $100,000?

# Hint: You may wish to compare the difference between the start and end date of the salaries contracts.

SELECT

COUNT(\*)

FROM

salaries

WHERE

salary >= 100000

AND DATEDIFF(to\_date, from\_date) > 365;

# Exercise 8

# Create a trigger that checks if the hire date of an employee is higher than the current date. If true, set this date to be the current date. Format the output appropriately (YY-MM-DD).

# Extra challenge: You may try to declare a new variable called 'today' which stores today's data, and then use it in your trigger!

# After creating the trigger, execute the following code to see if it's working properly.

/\*

INSERT employees VALUES ('999904', '1970-01-31', 'John', 'Johnson', 'M', '2025-01-01');

SELECT

\*

FROM

employees

ORDER BY emp\_no DESC;

\*/

DROP TRIGGER IF EXISTS trig\_hire\_date;

DELIMITER $$

CREATE TRIGGER trig\_hire\_date

BEFORE INSERT ON employees

FOR EACH ROW

BEGIN

DECLARE today date;

SELECT date\_format(sysdate(), '%Y-%m-%d') INTO today;

IF NEW.hire\_date > today THEN

SET NEW.hire\_date = today;

END IF;

END $$

DELIMITER ;

# Exercise 9

# Define a function that retrieves the largest contract salary value of an employee. Apply it to employee number 11356.

# Also, what is the lowest salary value per contract of the same employee? You may want to create a new function that will deliver this number to you. Apply it to employee number 11356 again.

# Feel free to apply the function to other employee numbers as well.

DROP FUNCTION IF EXISTS f\_highest\_salary;

DELIMITER $$

CREATE FUNCTION f\_highest\_salary (p\_emp\_no INTEGER) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE v\_highest\_salary DECIMAL(10,2);

SELECT

MAX(s.salary)

INTO v\_highest\_salary FROM

employees e

JOIN

salaries s ON e.emp\_no = s.emp\_no

WHERE

e.emp\_no = p\_emp\_no;

RETURN v\_highest\_salary;

END$$

DELIMITER ;

SELECT f\_highest\_salary(11356);

DROP FUNCTION IF EXISTS f\_lowest\_salary;

DELIMITER $$

CREATE FUNCTION f\_lowest\_salary (p\_emp\_no INTEGER) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE v\_lowest\_salary DECIMAL(10,2);

SELECT

MIN(s.salary)

INTO v\_lowest\_salary FROM

employees e

JOIN

salaries s ON e.emp\_no = s.emp\_no

WHERE

e.emp\_no = p\_emp\_no;

RETURN v\_lowest\_salary;

END$$

DELIMITER ;

SELECT f\_lowest\_salary(10356);

# Exercise 10

# Based on the previous example, you can now try to create a function that accepts also a second parameter which would be a character sequence.

# Evaluate if its value is 'min' or 'max' and based on that retrieve either the lowest or the highest salary (using the same logic and code

# from Exercise 9). If this value is a string value different from ‘min’ or ‘max’, then the output of the function should return

# the difference between the highest and the lowest salary.

DROP FUNCTION IF EXISTS f\_salary;

DELIMITER $$

CREATE FUNCTION f\_salary (p\_emp\_no INTEGER, p\_min\_or\_max varchar(10)) RETURNS DECIMAL(10,2)

DETERMINISTIC

BEGIN

DECLARE v\_salary\_info DECIMAL(10,2);

SELECT

CASE

WHEN p\_min\_or\_max = 'max' THEN MAX(s.salary)

WHEN p\_min\_or\_max = 'min' THEN MIN(s.salary)

ELSE MAX(s.salary) - MIN(s.salary)

END AS salary\_info

INTO v\_salary\_info FROM

employees e

JOIN

salaries s ON e.emp\_no = s.emp\_no

WHERE

e.emp\_no = p\_emp\_no;

RETURN v\_salary\_info;

END$$

DELIMITER ;

select employees.f\_salary(11356, 'min');

select employees.f\_salary(11356, 'max');

select employees.f\_salary(11356, 'maxxx');