SELECT \* FROM hr.employees;

-- Задача 1: Напишете и тествайте следните заявки :

-- 1. Write a SQL query to find the average salary in the "Sales" department. Use AVG().

SELECT d.department\_name AS 'Sales department', avg(salary) AS 'Average salary' FROM hr.employees AS e

JOIN departments AS d USING (department\_id)

WHERE department\_name = 'Sales';

-- 2. Write a SQL query to find the number of employees in the "Sales" department. Use COUNT(\*).

SELECT d.department\_name AS 'Sales department', COUNT(employee\_id) FROM employees

JOIN departments AS d USING (department\_id)

WHERE d.department\_name = 'Sales';

-- 3. Write a SQL query to find the number of all locations where the company has an office.

SELECT COUNT(location\_id) AS 'Number of locations' FROM locations;

-- 4. Write a SQL query to find the number of all departments that has manager.

SELECT department\_name FROM departments

WHERE manager\_id IS NOT NULL;

-- 5. Write a SQL query to find the number of all departments that has no manager.

SELECT department\_name FROM departments

WHERE manager\_id IS NULL;

-- 6. Write a SQL query to find all departments' names and the average salary for each of them.

SELECT department\_name, AVG(salary) AS 'Average salary' FROM departments

JOIN employees USING (department\_id)

GROUP BY department\_name;

-- 7. Write a SQL query to find the count of all employees in each department. Display the name, location and number of employees for each department.

SELECT d.department\_name, l.city, COUNT(e.employee\_id) AS 'Employee count' FROM locations AS l

JOIN departments AS d USING (location\_id)

JOIN employees AS e USING (department\_id)

GROUP BY d.department\_name;

-- 8. Write a SQL query to find for each department and for each manager the count of all corresponding employees.

SELECT d.department\_name, COUNT(e.employee\_id), CONCAT(m.first\_name, ' ', m.last\_name) AS 'Manager name',

COUNT(e.employee\_id) AS 'Number of employees' FROM departments AS d

JOIN employees AS e USING(department\_id)

JOIN employees AS m ON e.manager\_id = m.employee\_id

GROUP BY d.department\_name, e.manager\_id;

-- 9. Write a SQL query to find all managers that have exactly 5 employees. Display their names and the name and location of their department.

SELECT CONCAT(m.first\_name, ' ', m.last\_name) AS 'Manager name', l.city, department\_name, COUNT(\*) AS EmployeesToHandle

FROM employees m

JOIN departments d USING (department\_id)

JOIN locations AS l USING (location\_id)

GROUP BY m.manager\_id

HAVING COUNT(\*) = 5;

-- 10. Write a SQL query to find the total number of employees for each region.

SELECT region\_name, COUNT(employee\_id) FROM employees

JOIN departments USING (department\_id)

JOIN locations USING (location\_id)

JOIN countries USING (country\_id)

JOIN regions USING (region\_id)

GROUP BY region\_id;

-- 11. Write a SQL query to find for each department and for each job title the total number of employees.

SELECT d.department\_name, job\_title, COUNT(e.employee\_id) FROM employees e

JOIN departments AS d USING(department\_id)

JOIN jobs USING (job\_id)

GROUP BY department\_name, job\_title;

-- 12. Write a SQL query to find the names and salaries of the employees that take the minimal salary in the company. Use nested SELECT statement.

SELECT CONCAT(e.first\_name,' ', e.last\_name) AS 'Employee Name', salary FROM employees e

WHERE salary = (SELECT MIN(min\_salary) FROM jobs);

-- 13. Write a SQL query to find the names and salaries of the employees that get a salary that is up to 10% higher than the minimal salary for the company.

SELECT CONCAT(e.first\_name,' ', e.last\_name) AS 'Employee Name', salary FROM employees e

WHERE salary = (SELECT MIN(min\_salary)\*1.1 FROM jobs);

-- 14. Write a SQL that displays all departments and the highest salary for each department along with the name of the employee that takes it. If multiple employees in the same department have highest salary, display the first of them.

SELECT department\_name, MAX(salary), CONCAT(e.first\_name,' ', e.last\_name) AS Employee\_Name FROM employees e

JOIN departments USING (department\_id)

GROUP BY department\_name;

-- 15. Write a SQL query to find the names of all employees whose last name is exactly 5 characters long.

SELECT CONCAT(e.first\_name,' ', e.last\_name) AS Employee\_Name FROM employees e

WHERE length(e.last\_name) = 5;

-- 16. Write a SQL query to find the names of all employees whose first name and last name start with the same letter.

SELECT CONCAT(e.first\_name,' ', e.last\_name) AS Employee\_Name FROM employees e

WHERE substring(e.first\_name,1,1) = substring(e.last\_name,1,1);

-- 17. Display all departments names and their manager's name. For departments without manager display "(No manager)".

SELECT department\_name, if((CONCAT(e.first\_name,' ', e.last\_name) IS NOT NULL) , CONCAT(e.first\_name,' ', e.last\_name), 'No manager') AS Manager\_name FROM employees e

RIGHT JOIN departments USING(department\_id);

-- 18. Display all employees along with their number of directly managed people.

-- For employees not managing anybody display "Just an employee". For employees managing only 1 employee display "Junior manager". .

SELECT CONCAT(e.first\_name,' ',e.last\_name) AS Employee\_name, CASE CountOfEmployees WHEN 1 THEN 'Junior manager'

WHEN CountOfEmployees THEN CountOfEmployees

ELSE 'Just an employee' END AS Manager\_Status FROM employees AS e

LEFT JOIN (SELECT COUNT(\*) as CountOfEmployees, employee\_id FROM employees e

GROUP BY manager\_id) AS m ON e.employee\_id = m.employee\_id

ORDER BY Manager\_Status;

-- 19. Write a SQL query to print the current date and time in the format " hour:minutes:seconds day-month-year".

-- Display also the date coming after a week.

SELECT timestamp(curdate());

SELECT date\_format(current\_timestamp(),'%k:%i:%s %Y-%m-%d') AS Current\_TimeStamp\_Formatted;

-- 20. Write a SQL statement to create a table USERS. Users should have username, password,

-- full name and last login time. Choose appropriate data types for the fields of the table.

-- Define a primary key column with a primary key constraint. Define a trigger to automatically fill the full name column value before inserting a record.

CREATE TABLE users (

username varchar(30) NOT NULL PRIMARY KEY,

password varchar(30) NOT NULL,

full\_name varchar(30) NOT NULL,

last\_login\_time datetime NOT NULL)

DELIMITER $$

DROP TRIGGER IF EXISTS hr.users\_BEFORE\_INSERT$$

USE `hr`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `hr`.`users\_BEFORE\_INSERT` BEFORE INSERT ON `users` FOR EACH ROW

BEGIN

SET NEW.FULL\_NAME = 'Full name';

SET NEW.PASSWORD = 'PassWord';

SET NEW.last\_login\_time = current\_timestamp();

END$$

DELIMITER ;

INSERT INTO users (username,full\_name) values ('parola1', 'Ivan Dimitrov');

-- 21. Write a SQL statement to create a view that displays the users from the USERS table that have been in the system today.

-- Test if the view works correctly.

CREATE OR REPLACE VIEW today\_users AS

SELECT password, last\_login\_time

FROM users

WHERE password = 'PassWord';

SELECT \* FROM today\_users;

-- 22. Write a SQL statement to create a table GROUPS. Groups should have unique name (use unique constraint).

CREATE TABLE GROUPS (

name VARCHAR(30) NOT NULL UNIQUE

);

ALTER TABLE `hr`.`groups`

ADD COLUMN `id` VARCHAR(45) NOT NULL AFTER `name`,

ADD PRIMARY KEY (`id`);

SELECT \* FROM GROUPS;

-- 23. Write a SQL statement to add a column GROUP\_ID to the table USERS.

-- Fill some data in this new column and as well in the GROUPS table. Write a SQL statement to add a foreign

-- key constraint between tables USERS and GROUPS.

ALTER TABLE `hr`.`USERS`

ADD COLUMN `GROUP\_ID` VARCHAR(45) NOT NULL after `username`;

INSERT INTO GROUPS (id, name) values (1,'Ivan');

INSERT INTO GROUPS (id, name) values (2,'Petar');

INSERT INTO USERS (username, GROUP\_ID) VALUES(23,'58');

INSERT INTO USERS (username, GROUP\_ID) VALUES(27,25);

INSERT INTO GROUPS (id, name) values (3,'Ivo');

INSERT INTO GROUPS (id, name) values (4,'Nadia');

INSERT INTO GROUPS (id, name) values (5,'Filip');

UPDATE USERS SET GROUP\_ID = 1 WHERE username = '27';

UPDATE USERS SET GROUP\_ID = 2 WHERE username = '23';

UPDATE USERS SET GROUP\_ID = 3;

UPDATE USERS SET GROUP\_ID = 4

WHERE username = 'UserName';

UPDATE USERS SET GROUP\_ID = 3

WHERE username = 'parola';

UPDATE USERS SET GROUP\_ID = 5

WHERE username = 'parola1';

ALTER TABLE users MODIFY GROUP\_ID INTEGER;

ALTER TABLE `hr`.`groups`

CHANGE COLUMN `id` `id` INT NOT NULL ;

ALTER TABLE `hr`.`users`

CHANGE COLUMN `GROUP\_ID` `GROUP\_ID` INT NOT NULL ;

ALTER TABLE `hr`.`users`

ADD CONSTRAINT `GROUP\_ID`

FOREIGN KEY (`GROUP\_ID`)

REFERENCES `hr`.`groups` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION;

-- 24. Write SQL statements to insert several records in the USERS and GROUPS tables.

INSERT INTO groups (name, id) VALUES ('Vasko', 6);

INSERT INTO groups (name, id) VALUES ('Martin', 7);

INSERT INTO users (username, group\_id) VALUES ('VaskoParala', 6);

INSERT INTO users (username, group\_id) VALUES ('MartoParala', 7);

-- 25. Write SQL statements to insert in the USERS table the names of all employees from the employees table.

-- Combine the first and last names as a full name. For username use the email column from employees. Use blank password.

DELETE FROM `groups` WHERE `id` > 0;

INSERT INTO groups (name,id) SELECT CONCAT(first\_name, ' ', last\_name), employee\_id FROM employees;

INSERT INTO users (full\_name, username, group\_id) SELECT CONCAT(first\_name, ' ', last\_name), email, employee\_id FROM employees;

INSERT INTO users (password) VALUES('');

truncate users;

truncate groups;

-- 26. Run the above 10 times to generate enough testing data for the USERS table.

-- 27. Write a SQL statement that changes the password to NULL for all USERS that have not been in the system since 10.03.2006.

-- Select table data to see the changes.

UPDATE users SET password = null WHERE last\_login\_time < '2006-03-10';

UPDATE users SET password = null WHERE last\_login\_time < '2017-10-01';

-- 28. Write a SQL statement that deletes all users without passwords (NULL or empty password).

-- Select table data to see the changes.

DELETE FROM users WHERE password is NULL;

SELECT \* FROM users;

-- 29. Write a SQL query to list all users whose username starts with 's' and the number of groups for each of them.

SELECT username, COUNT(\*) FROM users

GROUP BY GROUP\_ID

HAVING username LIKE 's%';

-- 30. Define table WORKHOURS to store work reports for each employee (date, task, hours, comments).

CREATE TABLE WORKHOURS (

date date,

task varchar(30),

hours int,

comments varchar(255));

ALTER TABLE workhours ADD COLUMN id int auto\_increment PRIMARY KEY;

SELECT \* FROM WORKHOURS;

-- 31. Define foreign key between the tables WORKHOURS and EMPLOYEE. Add additional column in the employee table if needed.

ALTER TABLE employees ADD COLUMN workhours\_id int;

ALTER TABLE `hr`.`employees`

ADD CONSTRAINT `workhours\_id`

FOREIGN KEY (`workhours\_id`)

REFERENCES `hr`.`workhours` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION;

-- 32. Write several SQL statements to fill some data in the WORKHOURS table.

INSERT INTO workhours (date, task, hours, comments) values (current\_date(), 'do homework', current\_time(), 'making progress');

INSERT INTO workhours (date, task, hours, comments) values ( DATE\_ADD(curdate() , INTERVAL 2 DAY), 'do homework1', current\_time(), 'making progress');

INSERT INTO workhours (date, task, hours, comments) values ( DATE\_ADD(curdate() , INTERVAL 5 DAY), 'do homework1', current\_time(), 'making progress');

-- 33. Write a SQL query to find all the average work hours per week for each country.

SELECT country\_name, hours FROM workhours

JOIN employees ON workhours\_id = workhours.id

JOIN departments USING(department\_id)

JOIN locations USING(location\_id)

JOIN countries USING(country\_id)

GROUP BY country\_id;