Applied Databases

Topic 3 Exercise Sheet

Description of the database.

The database consists of two tables, *employees* and *salaries*.

The *employees* table is self-explanatory.

Salaries	
Attribute	Description
Emp_no	The Employee Number
Salary	The Employee's salary
From_date	The date the employee moved to this salary
To_date	The date the employee left this salary. If this date is 9999-01-01 it can be assumed that the employee is still on the salary

As an employee (emp_no) can have more than 1 salary the primary key of the table cannot be emp_no on its own and so is emp_no and from_date.

1. Get employees.sql from Moodle and import it into MySQL.

2. Print out the *emp_no*, *first_name* and a capitalised version of the employees *last_name*, using the same column names that are in the table for the first 10 employees returned from the database.

3. Sort the *employees* table based on:
   - The length of *last_name*
   - Alphabetical order of *last_name*
   - The length of *first_name*
   - Alphabetical order of *first_name*

4. Show all details of the first 10 employees returned from the database and an extra column called *Initials* that shows the employee's initials.

5. Show all details of all Females born in the 1950s and hired between September 1st 1988 and February 28th 1991.

6. Show the average salary from the *salaries* table formatted to two decimal places.
   E.g. 12345.6789 should become 12,345.68.

7. Show the *emp_no* and average salary for each employee formatted to two decimal places.

8. Show the *emp_no* and maximum salary for each employee formatted to two decimal places.

9. Show the *emp_no* and average salary formatted to two decimal places for the following employee numbers: 10001, 10021, 10033 and 10087.
   But only include in the average calculation salaries greater than 80,000.

10. Show the *emp_no* and average salary rounded to the nearest whole number only for average salaries greater than 90,000.

11. Show the following details, in the following order, for the first 15 employees, in *emp_no* order:
    ID, Title, Name, Surname, Gender.
    Title should be “Mr.” if the employee is Male, and “Ms.” if the employee is female.

12. Show the following details *emp_no*, the maximum salary for each employee, and the tax bracket the employee's maximum salary is in (Tax Bracket).
    Tax brackets are defined as follows:

Max Salary	Tax Bracket
Under 40,000	30%
Under 60,000	40%
Under 80,000	50%
Over 80,000	60%

```
select emp_no, max(salary),
CASE
  WHEN max(salary) < 40000 THEN "30%"
  WHEN max(salary) < 60000 THEN "40%"
  WHEN max(salary) < 80000 THEN "50%"
  ELSE "60%"
END as "Tax Bracket"
from salaries
group by emp_no
order by max(salary);
```

13. Show all details from the salaries table as well as a column entitled "Time" which states "Under 1 yr" if the employee has been on a particular salary for less than 365 days, otherwise states "Over 1 yr".

```
select *,
IF(datediff(to_date, from_date)<365,"Under 1 yr", "Over 1 yr")
as time
from salaries;
```

14. Using a function show all columns from the employees table, and a column entitled "Age" which is the age the employee was when he or she was hired. The age should be rounded to 1 digit after the decimal place.
    For example, employee 10001 was 32.8 years old when he was hired.
    HINT: Don't for get to change the delimiter when writing the function and change it back to a semi-colon when the function is written.

```
create function getage(d1 date, d2 date)
returns float(5,1)
deterministic
begin
  return round(datediff(d2,d1)/365,1);
end
//

select *, getage(birth_date, hire_date) as Age
from employees;
```

15. Write a procedure that takes two parameters, one representing a year and the other a month.
    The procedure should return all employees hired in specified year and month.

```
create procedure hires(y integer, m integer)
deterministic
begin
  select * from employees where year(hire_date) = y
  and month(hire_date) = m;
end
```

16. Rewrite the above procedure so that if the month parameter is NULL the procedure returns all employees hired in the specified year.
    If the month is not NULL, the procedure works as it did previously.
    HINT: To call a procedure with a NULL value for month (assuming in this case month is the second parameter) *procedure_name(1985, NULL)*.
    To check if a parameter, e.g. m, is NULL say *IF M IS NULL THEN*
    To check if a parameter, e.g. m, is not NULL say *IF M IS NOT NULL THEN.*

```
create procedure hires(y integer, m integer)
deterministic
begin
  if m is null then
     select * from employees where year(hire_date) = y;
  else
     select * from employees where year(hire_date) = y
     and month(hire_date) = m;
  end if;
end
```