**Nombre y apellidos: Alejandro Rodríguez López**

1. **Escribir las siguientes consultas como parte de una función, y escribir la instrucción que la invoque.**

**2.1** **Escribir una función denominada getInfoGrados() que liste el nombre y el número de estudiantes del grado que tenga el menor número de estudiantes.**

create function getInfoGrados ()

returns table (grado\_nombre varchar(100), numero\_estudiantes smallint) as $$

begin

return query select gra.grado\_nombre, gra.numero\_estudiantes from grado gra where

gra.numero\_estudiantes= (select min(g2.numero\_estudiantes) from grado g2);

end;

$$ language plpgsql;

**2.2.** **Amplia la consulta anterior, para ello escribe una función denominada getAulas() que liste solo aquellas aulas con un número de profesores mayor que 2.**

create function getAulas()

returns table (aula\_nombre varchar(25), cantidad bigint) as $$

begin

return query

select au.aula\_nombre, count(\*)

from aula au, profesor pro, modulo\_profesor\_aula mta

where au.aula\_id=mta.aula\_id and mta.profesor\_id=pro.profesor\_id

group by(au.aula\_id)

having count(pro.profesor\_id) >2;

if not found

then raise exception ‘No hay ningún aula con más de 2 profesores’;

end if;

end;

$$ language plpgsql;

* 1. **Escribe una función denominada getGradoMinEstudiantes() que presente el nombre del grado con el número más bajo de estudiantes. Recuerda que min(count(\*)) no es posible.**

create or replace function getGradoMinEstudiantes()

returns table (grado\_nombre varchar(100), cantidad bigint) as $$

begin

return query

select gra.grado\_nombre, count(\*) from grado gra inner join

estudiante\_grado\_modulo using(grado\_id) inner join modulo mod

using(modulo\_id) group by grado\_id having (count(\*)) <= all

(select count(\*) from grado gra inner join estudiante\_grado\_modulo using

(grado\_id) inner join modulo using(modulo\_id) group by gra.grado\_id);

end;

$$ language plpgsql;

**3. Convierte las siguientes consultas en procedimientos, y escribe la sentencia correspondiente que lo invoque.**

**3.1.** **Escribe un procedimiento denominado getcapacidadtotalaula que devuelva por parámetro la capacidad total entre todas las aulas.**

create procedure getCapacidadTotalAula(inout nAsientos int) as $$

begin

SELECT SUM(capacidad)

INTO nAsientos

FROM aula;

end;

$$ language plpgsql;

**3.2. Escribe un procedimiento denominado getEstudiantesIngenieriaIndustrial() que presente el nombre y los apellidos de aquellos estudiantes que estén estudiando ‘Ingenieria Quimica Industrial’, y también**

create procedure getEstudiantesIngenieriaIndustrial() as $$

declare r record;

begin

for r in

select estudiante\_nombre, estudiante\_apellidos from estudiante est

inner join estudiante\_grado\_modulo egm using(estudiante\_id)

inner join grado gra using(grado\_id)

where lower(gra.grado\_nombre)= 'Ingenieria Quimica Industrial'

union

select estudiante\_nombre, estudiante\_apellidos from estudiante est where

est.erasmus=true

loop

raise notice '(%, %)', r.estudiante\_nombre, r.estudiante\_apellidos;

end loop;

end;

$$ language plpgsql;

**3.3.** **Escribe un procedimiento denominado getProfesoresComputacionNoAlgoritmia(n int) que presente el nombre y los apellidos de los ‘n’ primeros profesores que pertenezcan al departamento ‘Ciencias de la Computacion’, pero sin tener en cuenta aquellos que imparten el módulo de ‘Algoritmia’.**

create or replace procedure getProfesoresComputacionNoAlgoritmia (n int) as $$

declare r record;

begin

for r in

(select profesor\_nombre, profesor\_apellidos from profesor pro

inner join departamento dep using(departamento\_id)

where dep.departamento\_nombre= 'Ciencias de la Computacion'

except

select profesor\_nombre, profesor\_apellidos from profesor pro

inner join modulo\_profesor\_aula mpa using(profesor\_id)

inner join modulo mod using(modulo\_id)

where lower(mod.modulo\_nombre)='algoritmia')

LIMIT(n)

loop

raise notice '(%, %)', r.profesor\_nombre, r.profesor\_apellidos;

end loop;

end;

$$ language plpgsql;