Nombre y apellidos:

2. **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 or replace function getInfoGrados()
    returns table(gradoNum smallint, gradoNombre
varchar(30)) as $$
    begin
        return query
            select gra.numero_estudiantes,
gra.grado_nombre
            from grado as gra
            where numero_estudiantes = (select
min(numero_estudiantes) from grado);
    end;
$$ language plpgsql;
select getInfoGrados();
```

   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 or replace function getAulas() returns
table(nombre varchar(30), capacidad bigint) as $$
    begin
        return query
            select au.aula_nombre, count(*)
            from aula au, profesor pro,
modulo_profesor_aula mta
            where au.aula_id=mta.aula_id and
mta.profesor_id=pro.profesor_id
            group by(au.aula_id)
            having count(pro.profesor_id) >2;
    end;
$$ language plpgsql;
```

```
select getAulas();
```

2.3. 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(gradoNombre varchar(30), gradoNum
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;
select getGradoMinEstudiantes();
```

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 or replace procedure
getCapacidadTotalAula(inout numero_asientos int) as
$$
    begin
        select sum(capacidad) into numero_asientos
from aula;
```

```
    end;
$$ language plpgsql;
call getCapacidadTotalAula(0);
```

3.2. Escribe un procedimiento denominada getEstudiantesIngenieriaIndustrial() que presente el nombre y los apellidos de aquellos estudiantes que estén estudiando 'Ingenieria Quimica Industrial', y también aquellos que son de erasmus.

```
-- Procedimiento que devuelva una lista con los
nombres y apellidos de aquellos estudiantes
-- que estén cursando 'Ingeniería Quimica
Industrial' y también aquellos que estén en erasmus.
create or replace 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;
call getEstudiantesIngenieriaIndustrial();
-- Resultado:
```

```
-- NOTICE: Sara Prendes Pardo
-- NOTICE: Juan Prieto Vazquez
-- NOTICE: Pedro Gancedo Alvarez
-- NOTICE: Maria Alvarez Gomez
```

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'.

```
-- 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; i integer default 0;
    begin
        for i in 1..n loop
            select * into r from (
                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') as t
                offset i-1 limit 1;
            raise notice '(%, %)',
r.profesor_nombre, r.profesor_apellidos;
        end loop;
    end;
$$ language plpgsql;
call getProfesoresComputacionNoAlgoritmia(4);
-- Resultado:
-- NOTICE: (Jose, Garcia Fanjul)
-- NOTICE: (Pablo, Suarez-Otero Gonzalez)
-- NOTICE: (Maria Jose, Suarez Cabal)
-- NOTICE: (Cristian Augusto, Alonso)
```