Nombre y apellidos: Alejandro Rodríguez López

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 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;
```

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 (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;
```