AMPLIACIÓ EXERCICIS PLPGSQL

A. Moll

1. CREATE FUNCTION

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [, ...] ] )
    [ RETURNS rettype ]
  { LANGUAGE langname
    | IMMUTABLE | STABLE | VOLATILE
    | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
    |[EXTERNAL] SECURITY INVOKER | [EXTERNAL] SECURITY DEFINER
    | COST execution_cost
    | ROWS result_rows
    | SET conf_parameter {TO value | = value | FROM CURRENT}
    | AS 'definition'
    | AS 'obj_file', 'link_symbol'
  }•••
    [ WITH ( attribute [, ...] ) ]
```

🗣 *argmode* ➔ The mode of an argument: either `IN`, `OUT`, or `INOUT`. The default is `IN`.

🗣 The type of a column is referenced by writing ***tablename.columnname*%TYPE**. Using this feature can sometimes help make a function independent of changes to the definition of a table.

When there are `OUT` or `INOUT` parameters, the `RETURNS` clause can be omitted. If present, it must agree with the result type implied by the output parameters: `RECORD` if there are multiple output parameters, or the same type as the single output parameter.

1.1. Sentència Return

Toda función requiere la cláusula ***return*** *expresión.* La expresión será convertida automáticamente al tipo de dato especificado en la cabecera de la función mediante **returns**.

If you declared the function with output parameters ➔ Write just `RETURN` with no expression.

If you declared the function to return `void`, a `RETURN` statement can be used to exit the function early; but do not write an expression following `RETURN`.

1. Funció que lliste la taula de multiplicar del paràmetre numéric introduït.

2. Declara una seqüència que només genere números pars.

CREATE SEQUENCE . Creates a new sequence number generator
CREATE SEQUENCE *seqname*
[INCREMENT *increment*]
[MINVALUE *minvalue*]
[MAXVALUE *maxvalue*]
[START *start*]
[CACHE *cache*]
[CYCLE]

Create sequence Pregunta2 increment 2 start 2;

O bé

Create sequence Preguntaw increment 2 Minvalue 2;

3. Función que devuelva el **factorial** de un parámetro de entrada.

4. Divisió entera utilitzant una funció SQL recursiva

```
CREATE OR REPLACE FUNCTION DIV (int, int) RETURNS INT
AS $$ -- Definició recursiva de la funció divisió d'enters
```

```
  SELECT CASE
   WHEN $1<$2  THEN 0
   WHEN $1 >= $2 THEN 1 + DIV ($1-$2, $2)
  END;
$$ LANGUAGE sql IMMUTABLE;
```

5. Función ***LongCircunferencia*** que devuelva la longitud de una circunferencia cuyo radio le pasaremos como parámetro.

6. Función que devuelva si cierto número es primo

7. Función **SUMATORIO** que devuelva la suma desde 1 hasta el número introducido por teclado. Así si lanzamos Select sumatorio(4) ➔ 1+2+3+4 = 10 .

8. Escriure una funció per crear una taula Articles = {Codi, Nom, Preu, Data}.

```
CREATE OR REPLACE FUNCTION NovaTaula () RETURNS VOID
AS
$$
CREATE TABLE Articles   (
        Codi numeric(3) NOT NULL PRIMARY KEY,   Nom text, Preu numeric (7,2), Data date
) $$
LANGUAGE sql;
```

9. Escriure una funció que establisca l'estil de les dates en format europeu

```
CREATE OR REPLACE FUNCTION estilData () RETURNS VOID
AS
$$    SET DATESTYLE TO PostgreSQL,European;
$$ LANGUAGE sql;
```

10. Escriu una funció que escriga una cadena en ordre invertit.

11. Escriu una funció que ens diga si una cadena de caràcters es o no palíndrom.

```
SELECT  Palindrom('DABALE ARROZ A LA ZORRA EL ABAD');
```

12. Crea la función ***AumentarNotaATodos*** que incremente todas las notas en una cantidad que se introducirá por teclado. Tened la precaución de no sobrepasar los 10 puntos.

```
create or replace function AugmentarNota(in augment Notes.nota%type) returns void as
$$ /* Recordem que, per defecte, els paràmetres són  IN */
begin
 if augment is not null then
```

```
                        update notes
                        set nota=10
                        where coalesce(nota,0)+augment>10;

                        update notes
                        set nota=coalesce(nota,0)+augment
                        where coalesce(nota,0)+augment<=10;
/* En eixe ordre hem de fer els updates */
 end if;
end; /* No cal ficar  sentència  RETURN, com s'exigia en antigues versions del motor,  tot i que la
funció és tipus void */
$$
language plpgsql;
```

13. Función ***spread*** que pasa a mayúsculas y expanda, insertando un blanco en entre cada carácter, una cadena introducida como parámentro.

Exemple

Select spread('Hola mundo') ➔ H O L A M U N D O

14. Crea la función ***ElevarA*** que será un alias para la función ***power*** que lleva a cabo la exponenciación de números de doble precisión.

15. Función ***BuscaAlumne*** que al pasarle un número de expediente, conteste con el mensaje "***Este alumno sí existe***" o en caso contrario, "***El alumno que usted busca no está en la base de datos***".

```
create or replace function BuscaAlumne(Alumnes.Nexpd%TYPE) returns void as
$$
declare
        aux Alumnes.Nexpd%TYPE;
begin
        select nexpd into aux
        from alumnes
        where nexpd=$1;
        if not found then raise notice 'El alumno que usted busca no está en la
base de datos';
                else raise notice 'Este alumno sí existe';
        end if;
end;
$$
language plpgsql;
```

The second method to determine the effects of a command is to check the special variable named `FOUND`, which is of type `boolean`. `FOUND` starts out false within each PL/pgSQL function call. It is set by each of the following types of statements:

- A `SELECT INTO` statement sets `FOUND` true if a row is assigned, false if no row is returned.

- A `PERFORM` statement sets `FOUND` true if it produces (and discards) one or more rows, false if no row is produced.
- `UPDATE`, `INSERT`, and `DELETE` statements set `FOUND` true if at least one row is affected, false if no row is affected.
- A `FETCH` statement sets `FOUND` true if it returns a row, false if no row is returned.
- A `MOVE` statement sets `FOUND` true if it successfully repositions the cursor, false otherwise.
- A `FOR` statement sets `FOUND` true if it iterates one or more times, else false. This applies to all three variants of the `FOR` statement (integer `FOR` loops, record-set `FOR` loops, and dynamic record-set `FOR` loops). `FOUND` is set this way when the `FOR` loop exits; inside the execution of the loop, `FOUND` is not modified by the `FOR` statement, although it might be changed by the execution of other statements within the loop body.

`FOUND` is a local variable within each PL/pgSQL function; any changes to it affect only the current function.

2. TRIGGERS

2.1. Exercicis per obrir boca...

16. ¿Qué ocurre cuando dos o más triggers con el mismo evento se asocian a una misma tabla?.

17. ¿Puede una misma función asociarse a disparadores distintos?

18. ¿Qué sentido tienen las variables NEW y OLD en disparadores con cláusula FOR EACH STATEMENT?

19. Dissenya un trigger sobre una taula senzilleta que genere una clau ppal autoincrementativa.

Veurem dos solucions. Ambdos precissen la creació prèvia d'un generador o seqüència:

```
Create sequence comptador
```

Primera forma: En el propi *create table*

```
Create table prova
(codi integer primary key default nextval('comptador'),
descripcio varchar(10));
```

Segona forma: Mitjançant un trigger

Primer es declara la funció-trigger i després el propi trigger.

```
create or replace function funcio() returns opaque
/* Opaque és el tipus especial que les versions
7 de Postgres reserven per a les especials funcions
que son cridades pels disparadors*/
as
'
begin
new.codi:=nextval("comptador");
/* Després de patir prou m'he adonat que cal ficar dues
```

```
create trigger generaclau
before insert
on prova
for each row
execute procedure funcio();
```

cometes a cada costat de la seqüència*/ return new; /* Retornem new per exigència sintàctica*/ end; ' language plpgsql;	

2.2. Sobre la taula MiniAgenda…

20. Sobre la taula Miniagenda dissenya un trigger que faça el següent:

- Generació automàtica de la clau
- Dese la primera lletra del nom en majúscules i la resta en minúscules.
- Dese el cognom en majúscules.
- Dese automàticament la data d'inserció de la tupla
- Dese automàticament l'edat de la persona a partir de la seua data de naiximent.

```
CREATE TABLE MINIAGENDA
(CODI integer primary key,
 NOM VARCHAR(12) not null,
 COGNOM VARCHAR(12),
 EDAT integer,
 DATANAIXIMENT DATE,
 DATAINSERCIO DATE,
 TLF CHAR(9));
```