

EXERCICIS D'AULA

EA3

BBDD - 12/04/2024

Jashan

Exercici 1

```
CREATE OR REPLACE FUNCTION Escribe_No_Multiples(num INT)
RETURNS VOID AS $$
DECLARE
    i INT := 1;
BEGIN
    WHILE i <= num LOOP
        IF i % 3 = 0 THEN
            RAISE NOTICE '%', 'Singh';
        ELSE
            RAISE NOTICE '%', i;
        END IF;
        i := i + 1;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT Escribe_No_Multiples(30);
```

```
NOTICE: 1
NOTICE: 2
NOTICE: Singh
NOTICE: 4
NOTICE: 5
NOTICE: Singh
```

Exercici 2

```
CREATE OR REPLACE FUNCTION comprobar_longitud(cadena TEXT)
RETURNS TEXT AS
$$
DECLARE
    cadena_maj TEXT;
BEGIN
    IF LENGTH(cadena) > 10 THEN
        cadena_maj := UPPER(cadena);
        RETURN cadena_maj;
    ELSE
        RAISE EXCEPTION 'La longitud de la cadena ha de ser mayor que 10 caràcters';
    END IF;
END;
$$
LANGUAGE plpgsql;
```

```
select
comprovar_longitud('mira quina cadena de text');
```

```
select comprobar_longitud('hola');
```

	comprovar_longitud text
1	MIRA QUINA CADENA DE TEXT

```
ERROR: La longitud de la cadena ha de ser mayor que
10 caràcters
CONTEXT: PL/pgSQL function comprobar_longitud(text)
line 9 at RAISE
```

Exercici 3

```
CREATE OR REPLACE FUNCTION es_numero_primer(enter INTEGER, OUT es_primer
BOOLEAN) RETURNS BOOLEAN AS $$
DECLARE
    i INTEGER;
BEGIN
    IF enter <= 1 THEN
        RAISE EXCEPTION 'El nombre ha de ser major que 1';
    END IF;

    es_primer := TRUE;
    FOR i IN 2..sqrt(enter) LOOP
        IF enter % i = 0 THEN
            es_primer := FALSE;
            EXIT;
        END IF;
    END LOOP;

    IF es_primer THEN
        RAISE NOTICE 'El nombre % és un nombre primer', enter;
    ELSE
        RAISE NOTICE 'El nombre % no és un nombre primer', enter;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

select es_numero_primer(5);

NOTICE: El nombre 5 és un nombre primer

	es_numero_primer
	boolean

1	true
---	------

select es_numero_primer(12);

NOTICE: El nombre 12 no és un nombre primer

	es_numero_primer
	boolean

1	false
---	-------

Exercici 4

Tenia un parell de bbdd soltes, he ficat les dues taules en una d'elles. Hi ha un error amb les taules; les vaig ficar en plural en comptes de dir-les "departament" i "usuari".

```
CREATE TABLE departaments (  
  id SERIAL PRIMARY KEY,  
  nom VARCHAR(100) UNIQUE  
);  
  
CREATE TABLE usuaris (  
  id SERIAL PRIMARY KEY,  
  nom VARCHAR(100),  
  departament_id INT REFERENCES departaments(id)  
);
```

Fico dos departaments d'exemple.

```
154 INSERT INTO departaments (nom) VALUES ('Departament de Vendes');  
155 INSERT INTO departaments (nom) VALUES ('Departament de Recursos Humans');  
156  
157 select * from departaments;
```

Data Output Messages Notifications

	id [PK] integer	nom character varying (100)
1	1	Departament de Vendes
2	2	Departament de Recursos Humans

```
CREATE OR REPLACE FUNCTION inserir_usuari(nom_usuari VARCHAR,  
nom_departament VARCHAR)  
RETURNS VOID AS $$  
BEGIN  
  INSERT INTO usuaris (nom, departament_id)  
  VALUES (nom_usuari, (SELECT id FROM departaments WHERE nom =  
nom_departament));  
END;  
$$ LANGUAGE plpgsql;
```

```
160 SELECT inserir_usuari('Pepe', 'Departament de Vendes');  
161 SELECT inserir_usuari('Manolet', 'Departament de Recursos Humans');  
162  
163 select * from usuaris;  
164
```

Data Output Messages Notifications

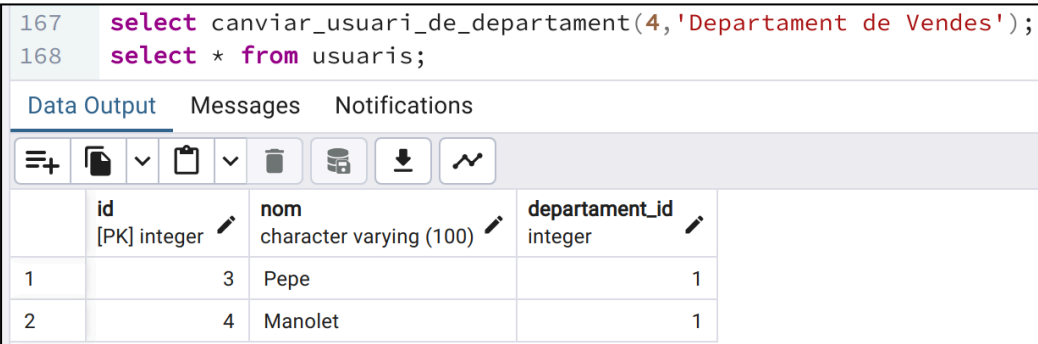
	id [PK] integer	nom character varying (100)	departament_id integer
1	3	Pepe	1
2	4	Manolet	2

Hem vist que la funció funciona, també ens fica automaticament l'id del departament (fa el select del id a la funció).

```
CREATE OR REPLACE FUNCTION eliminar_usuari(id_usuari INT)
RETURNS VOID AS $$
BEGIN
    DELETE FROM usuaris WHERE id = id_usuari;
END;
$$ LANGUAGE plpgsql;
```

Simplement faig un SELECT eliminar_usuari(3); i borra el usuari Pepe.

```
CREATE OR REPLACE FUNCTION canviar_usuari_de_departament(id_usuari INT,
nou_departament VARCHAR)
RETURNS VOID AS $$
BEGIN
    UPDATE usuaris SET departament_id = (SELECT id FROM departaments WHERE
nom = nou_departament)
    WHERE id = id_usuari;
END;
$$ LANGUAGE plpgsql;
```



	id [PK] integer	nom character varying (100)	departament_id integer	
1	3	Pepe	1	
2	4	Manolet	1	

Usuari Manolet canviat al primer departament perfectament.

```
CREATE OR REPLACE FUNCTION obtenir_usuaris_departament(nom_departament
VARCHAR)
RETURNS TABLE(nom_usuari VARCHAR) AS $$
BEGIN
    RETURN QUERY SELECT us.nom
    FROM usuaris us
    JOIN departaments dep ON us.departament_id = dep.id
    WHERE dep.nom = nom_departament;
END;
$$ LANGUAGE plpgsql;
```

En tenim dos a Vendes.

```
153 select obtenir_usuaris_departament('Departament de Vendes');
```

Data Output		Messages	Notifications
obtenir_usuaris_departament character varying			
1	Pepe		
2	Manolet		

I no en tenim cap a RRHH (vam canviar el dept. del Manolet)

```
153 select obtenir_usuaris_departament('Departament de Recursos Humans');
```

Data Output		Messages	Notifications
obtenir_usuaris_departament character varying			

Exercici 5

```
ALTER TABLE usuari  
ADD COLUMN edat INT,  
ADD COLUMN sexe CHAR(1),  
ADD COLUMN descripcio TEXT,  
ADD COLUMN data_naixement DATE;
```

```
INSERT INTO usuari (nom, departament_id, data_naixement)  
VALUES  
('Maria', 1, '2000-03-15'),  
('Heroïna', 2, '1995-07-20'),  
('Catalina', 1, '2005-11-10');
```

```
200 SELECT * FROM usuaris;
```

	id [PK] integer	nom character varying (100)	departament_id integer	edat integer	sexe character	descripcio text	data_naixement date
1	3	Pepe	1	[null]	[null]	[null]	[null]
2	4	Manolet	1	[null]	[null]	[null]	[null]
3	5	Maria	1	[null]	[null]	[null]	2000-03-15
4	6	Heroïna	2	[null]	[null]	[null]	1995-07-20
5	7	Catalina	1	[null]	[null]	[null]	2005-11-10

Fem servir la funció per saber l'edat dels tres

```
197 UPDATE usuaris
198 SET edat = calcular_edat(data_naixement);
199 SELECT * FROM usuaris;
```

	id [PK] integer	nom character varying (100)	departament_id integer	edat integer	sexe character	descripcio text	data_naixement date
1	3	Pepe	1	[null]	[null]	[null]	[null]
2	4	Manolet	1	[null]	[null]	[null]	[null]
3	5	Maria	1	24	[null]	[null]	2000-03-15
4	6	Heroïna	2	29	[null]	[null]	1995-07-20
5	7	Catalina	1	19	[null]	[null]	2005-11-10

Insereixo un usuari que sigui menor d'edat (la Josefina)

	id [PK] integer	nom character varying (100)	departament_id integer	edat integer	sexe character	descripcio text	data_naixement date
1	3	Pepe	1	[null]	[null]	[null]	[null]
2	4	Manolet	1	[null]	[null]	[null]	[null]
3	5	Maria	1	24	[null]	[null]	2000-03-15
4	6	Heroïna	2	29	[null]	[null]	1995-07-20
5	7	Catalina	1	19	[null]	[null]	2005-11-10
6	9	Josefina	1	[null]	[null]	[null]	2009-11-10

```
CREATE OR REPLACE FUNCTION esborrar_usuaris_menors_edat()
RETURNS VOID AS $$
BEGIN
    DELETE FROM usuaris WHERE calcular_edat(data_naixement) < 18;
END;
$$ LANGUAGE plpgsql;
```

208	SELECT	esborrar_usuaris_menors_edat();
209	select	* from usuaris;

Data Output	Messages	Notifications
-------------	----------	---------------

--	--	--	--	--	--	--

	id [PK] integer	nom character varying (100)	departament_id integer	edat integer	sexe character	descripcio text	data_naixement date
1	3	Pepe	1	[null]	[null]	[null]	[null]
2	4	Manolet	1	[null]	[null]	[null]	[null]
3	5	Maria	1	24	[null]	[null]	2000-03-15
4	6	Heroïna	2	29	[null]	[null]	1995-07-20
5	7	Catalina	1	19	[null]	[null]	2005-11-10

Ens esborra correctament el usuari que era menor d'edat.

Exercici 6 - Funció disparar

```
CREATE OR REPLACE FUNCTION disparar(cargador INT, projectil INT) RETURNS
BOOLEAN AS $$
BEGIN
    IF cargador = projectil THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
$$ LANGUAGE plpgsql;

select disparar(1,1);
```

Exercici 6 - Funció jugar_ruleta


```

CREATE OR REPLACE FUNCTION jugar_ruleta(jugadors TEXT[]) RETURNS TEXT AS
$$
DECLARE
    posicio_bala INT;
    posicio_lliure INT;
    jugador_actual TEXT;
    i INT := 1;
    num_jugadors INT := array_length(jugadors, 1);
BEGIN
    WHILE i <= num_jugadors LOOP
        jugador_actual := jugadors[i];
        posicio_bala := floor(random() * 6) + 1;
        posicio_lliure := floor(random() * 6) + 1;


        IF disparar(posicio_bala, posicio_lliure) THEN
            RETURN 'El perdedor és el jugador: ' || jugador_actual;
        ELSE
            RAISE NOTICE 'Jugador actual: % ha sobreviscut', jugador_actual;
        END IF;


        i := i + 1;
    END LOOP;

    RETURN 'No hi ha guanyadors, tots els jugadors han sobreviscut';
END;
$$ LANGUAGE plpgsql;

SELECT jugar_ruleta(ARRAY['Jugador 1', 'Jugador 2', 'Jugador 3', 'Jugador 4']);

```

	jugar_ruleta text	
1	El perdedor és el jugador: Jugador 2	

	jugar_ruleta text	
1	No hi ha guanyadors, tots els jugadors han sobreviscut	