



## **TALLER DE ADMINISTRACIÓN DE DATOS**

### **TALLER 4 - PostgreSQL**

#### **INTEGRANTES:**

Johan Sneider Albarracin Gómez - 160003901  
Edgar David Lozada González – 160004023  
Carlos Danilo Núñez Gil – 160004032

Ejecute el siguiente código:

```
alter table ENTREGA_ELEMENTOS
add DURACION NUMBER DEFAULT 0;

update ENTREGA_ELEMENTOS set DURACION = 30;

alter table EMPLEADOS
add CLAVE VARCHAR2(250) DEFAULT '---';
```

Table ENTREGA\_ELEMENTOS alterado.

20 filas actualizadas.

Table EMPLEADOS alterado.

Realice los siguientes ejercicios:

- Cree un bloque anónimo que visualice los números pares del 0 al 100.

```
1. DO $$
2. DECLARE
3. BEGIN
4.     FOR numero IN 0..100 LOOP
5.         IF MOD(numero, 2) = 0 THEN
6.             RAISE NOTICE 'NUMERO: %', numero;
7.         END IF;
8.     END LOOP;
9. END;
```



```
10. $$ LANGUAGE PLPGSQL;
```

```
NOTICE: NUMERO: 0  
NOTICE: NUMERO: 2  
NOTICE: NUMERO: 4  
NOTICE: NUMERO: 6  
NOTICE: NUMERO: 8  
NOTICE: NUMERO: 10  
NOTICE: NUMERO: 12  
NOTICE: NUMERO: 14  
NOTICE: NUMERO: 16  
NOTICE: NUMERO: 18  
NOTICE: NUMERO: 20  
NOTICE: NUMERO: 22  
NOTICE: NUMERO: 24  
NOTICE: NUMERO: 26  
NOTICE: NUMERO: 28  
NOTICE: NUMERO: 30
```

- Cree un bloque anónimo que visualice los números primos del 0 al 100.

```
1. DO $$  
2. DECLARE  
3.     primo BOOLEAN;  
4. BEGIN  
5.     FOR numero IN 0..100 LOOP  
6.         FOR divisor IN 2..numero LOOP  
7.             IF MOD(numero,divisor) = 0 AND numero!=divisor THEN  
8.                 primo := FALSE;  
9.                 EXIT;  
10.            ELSE  
11.                primo := TRUE;  
12.            END IF;  
13.        END LOOP;  
14.        IF primo = TRUE THEN  
15.            RAISE NOTICE 'NUMERO: %',numero;  
16.        END IF;  
17.    END LOOP;  
18. END;  
19. $$ LANGUAGE PLPGSQL;
```



```
NOTICE: NUMERO: 2
NOTICE: NUMERO: 3
NOTICE: NUMERO: 5
NOTICE: NUMERO: 7
NOTICE: NUMERO: 11
NOTICE: NUMERO: 13
NOTICE: NUMERO: 17
NOTICE: NUMERO: 19
NOTICE: NUMERO: 23
NOTICE: NUMERO: 29
NOTICE: NUMERO: 31
NOTICE: NUMERO: 37
NOTICE: NUMERO: 41
NOTICE: NUMERO: 43
NOTICE: NUMERO: 47
NOTICE: NUMERO: 53
```

- Cree un bloque anónimo que solicite dos números y permita conocer cuál de los dos números es mayor o indicar si es el caso si son iguales.

```
1. DO $$
2. DECLARE
3.   numero1 INTEGER := 7;
4.   numero2 INTEGER := 6;
5. BEGIN
6.   IF numero1 > numero2 THEN
7.     RAISE NOTICE '% ES MAYOR QUE %', numero1, numero2;
8.   ELSIF numero1 = numero2 THEN
9.     RAISE NOTICE '% ES IGUAL QUE %', numero1, numero2;
10.  ELSE
11.    RAISE NOTICE '% ES MAYOR QUE %', numero2, numero1;
12.  END IF;
13. END;
14. $$ LANGUAGE PLPGSQL;
```

```
NOTICE: 7 ES MAYOR QUE 6
DO
```

```
Query returned successfully in 157 msec
```

- Cree un bloque anónimo que ingrese 1000 registros a la siguiente tabla:



```
CREATE TABLE TMP_MISDATOS
(
  ID INTEGER PRIMARY KEY,
  DESCRIPCION VARCHAR2(30) NOT NULL
);
```

⚠️ **Nota:** El campo ID debe ser un número primo y el campo descripción debe tener la siguiente cadena de texto: 'El identificador es 2', donde el numero es el id que generaron.

CREATE TABLE

Query returned successfully in 66 msec.

```
1. DO $$
2. DECLARE
3.     primo BOOLEAN;
4.     descrip VARCHAR(30) := 'El identificador es';
5.     cantidad INTEGER := 1000;
6.     contador INTEGER := 1;
7.     numero INTEGER := 0;
8. BEGIN
9.     WHILE contador <= cantidad LOOP
10.         FOR divisor IN 2..numero LOOP
11.             IF MOD(numero, divisor) = 0 AND numero != divisor
12. THEN
13.                 primo := FALSE;
14.                 EXIT;
15.             ELSE
16.                 primo := TRUE;
17.             END IF;
18.         END LOOP;
19.         IF primo = TRUE THEN
20.             INSERT INTO tmp_misdatos(id, descripcion) VALUES
21. (numero, descrip || ' ' || numero);
22.             contador := contador+1;
23.             numero := numero+1;
24.         ELSE
25.             numero := numero+1;
26.         END IF;
27.     END LOOP;
28.     COMMIT;
29. END;
30. $$ LANGUAGE PLPGSQL;
```



id [PK] integer	descripcion character varying (30)
2	El identificador es 2
3	El identificador es 3
5	El identificador es 5
7	El identificador es 7
11	El identificador es 11
13	El identificador es 13
17	El identificador es 17
19	El identificador es 19

- Cree un bloque anónimo que actualice la fecha de devolución de la tabla ENTREGA\_ELEMENTOS, donde la duración (días) de cada elemento esta en el campo duracion.

```
1. DO $$  
2. BEGIN  
3.     UPDATE entrega_elementos SET fecha_devolucion=fecha+duracion;  
4.     COMMIT;  
5. END;  
6. $$ LANGUAGE PLPGSQL;
```

numero character varying (15)	devolutivo character varying (2)	fecha_devolucion date	empleado [PK] integer	fecha date	duracion integer
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30
	N	2009-09-14	40987267	2009-08-15	30

- Cree un procedimiento para el punto 1 con el nombre de: pares.



```
1. CREATE OR REPLACE PROCEDURE pares()  
2. AS $$  
3. BEGIN  
4.     FOR numero IN 0..100 LOOP  
5.         IF MOD(numero, 2) = 0 THEN  
6.             RAISE NOTICE 'NUMERO: %', numero;  
7.         END IF;  
8.     END LOOP;  
9. END;  
10. $$ LANGUAGE PLPGSQL;  
11. CALL pares();
```

```
NOTICE: NUMERO: 0  
NOTICE: NUMERO: 2  
NOTICE: NUMERO: 4  
NOTICE: NUMERO: 6  
NOTICE: NUMERO: 8  
NOTICE: NUMERO: 10  
NOTICE: NUMERO: 12  
NOTICE: NUMERO: 14  
NOTICE: NUMERO: 16  
NOTICE: NUMERO: 18  
NOTICE: NUMERO: 20  
NOTICE: NUMERO: 22  
NOTICE: NUMERO: 24  
NOTICE: NUMERO: 26  
NOTICE: NUMERO: 28  
NOTICE: NUMERO: 30
```

- Cree un procedimiento para el punto 2 con el nombre de: primos.

```
1. CREATE OR REPLACE PROCEDURE primos()  
2. AS $$  
3. DECLARE  
4.     primo BOOLEAN;  
5. BEGIN  
6.     FOR numero IN 0..100 LOOP  
7.         FOR divisor IN 2..numero LOOP  
8.             IF MOD(numero, divisor) = 0 AND numero != divisor THEN  
9.                 primo := FALSE;  
10.                EXIT;  
11.            ELSE  
12.                primo := TRUE;  
13.            END IF;
```



```
14.          END LOOP;  
15.          IF primo = TRUE THEN  
16.              RAISE NOTICE 'NUMERO: %',numero;  
17.          END IF;  
18.      END LOOP;  
19.  END;  
20.  $$ LANGUAGE PLPGSQL;  
21.  CALL primos();
```

```
NOTICE: NUMERO: 2  
NOTICE: NUMERO: 3  
NOTICE: NUMERO: 5  
NOTICE: NUMERO: 7  
NOTICE: NUMERO: 11  
NOTICE: NUMERO: 13  
NOTICE: NUMERO: 17  
NOTICE: NUMERO: 19  
NOTICE: NUMERO: 23  
NOTICE: NUMERO: 29  
NOTICE: NUMERO: 31  
NOTICE: NUMERO: 37  
NOTICE: NUMERO: 41  
NOTICE: NUMERO: 43  
NOTICE: NUMERO: 47  
NOTICE: NUMERO: 53
```

- Cree un procedimiento para el punto 3 con el nombre de: comparar\_numeros, este procedimiento debe recibir por parámetros los dos números a comparar.

```
1. CREATE OR REPLACE PROCEDURE comparar_numeros(numero1 INTEGER, numero2 INTEGER)  
2. AS $$  
3. BEGIN  
4.     IF numero1 > numero2 THEN  
5.         RAISE NOTICE '% ES MAYOR QUE %',numero1,numero2;  
6.     ELSIF numero1 = numero2 THEN  
7.         RAISE NOTICE '% ES IGUAL QUE %',numero1,numero2;  
8.     ELSE  
9.         RAISE NOTICE '% ES MAYOR QUE %',numero2,numero1;  
10.    END IF;  
11. END;  
12. $$ LANGUAGE plpgsql;  
13. CALL comparar_numeros(4,2)
```



```
NOTICE:  4 ES MAYOR QUE 2  
CALL
```

```
Query returned successfully in 152 msec.
```

- Cree un procedimiento para actualizar la columna email de la tabla empleados con la nueva dirección de correo asignada por la empresa. La nueva dirección asignada debe estar conformada por:
  - a) La inicial del primer nombre + el primer apellido + el segundo apellido + unillanos.edu.co. Ejemplo: Ricardo Andrés Sandoval Morales -> rsandovalmorales@unillanos.edu.co Además, debe generar la clave del correo la cual debe estar conformada por: Inicial del primer nombre + inicial del segundo nombre + la identificación.
  - b) Debe validar que no se repitan las direcciones, si se repita alguna, debe agregar un número al final del nombre de la dirección de correo, el número es un consecutivo que debe iniciar en 1.

Función para buscar correos repetidos:

```
1. CREATE OR REPLACE FUNCTION buscar_email_repetido(correo  
empleados.email%TYPE,id empleados.identificacion%TYPE)  
2. RETURNS BOOLEAN AS $$  
3. DECLARE  
4.     contador INTEGER := 0;  
5. BEGIN  
6.     SELECT COUNT(emp.email) INTO contador FROM empleados AS emp W  
HERE emp.email = correo AND emp.identificacion != id;  
7.     IF contador > 0 THEN  
8.         RETURN TRUE;  
9.     ELSE  
10.        RETURN FALSE;  
11.    END IF;  
12.    END;  
13.    $$ LANGUAGE plpgsql;
```

Procedimiento para ingresar la nueva dirección de correo:

```
1. CREATE OR REPLACE PROCEDURE actualizar_correos()  
2. AS $$  
3. DECLARE
```





```
4.      cur_empleados CURSOR FOR SELECT nombre_1, nombre_2,
      apellido_1, apellido_2, identificacion FROM empleados;
5.      correo_nuevo empleados.email%TYPE;
6.      clave_nueva empleados.clave%TYPE;
7.      dominio VARCHAR(17) := '@UNILLANOS.EDU.CO';
8.      contador INTEGER;
9. BEGIN
10.         -- Creamos un cursor para recorrer la tabla de
      empleados
11.         FOR empleado IN cur_empleados LOOP
12.             -- Generamos la nueva direccion de correo
13.             correo_nuevo
      := CONCAT(SUBSTR(empleado.nombre_1,1,1),empleado.apellido_1,emple
      ado.apellido_2,dominio);
14.             -- Generamos la clave del correo
15.             clave_nueva
      := CONCAT(SUBSTR(empleado.nombre_1,1,1),SUBSTR(empleado.nombre_2,
      1,1),empleado.identificacion);
16.             --Validamos que no se repita la direccion de correo
17.             WHILE buscar_email_repetido(correo_nuevo,empleado.i
      dentificacion) = TRUE LOOP
18.                 contador := 1;
19.                 correo_nuevo
      := CONCAT(SUBSTR(empleado.nombre_1,1,1),empleado.apellido_1,emple
      ado.apellido_2,contador,dominio);
20.                 contador := contador + 1;
21.             END LOOP;
22.             -- Actualizamos la direccion de correo y la clave
      del correo en la tabla de empleados
23.             UPDATE empleados SET email = correo_nuevo,
      clave = clave_nueva WHERE identificacion = empleado.identificacio
      n;
24.             COMMIT;
25.         END LOOP;
26.     END;
27.     $$ LANGUAGE plpgsql;
28.     CALL actualizar_correos();
```



identificacion integer	nombre_completo text	email character varying (100)	clave character varying
40987267	ANDREA CASTAÑO MARTINEZ	ACASTAÑO MARTINEZ1@UNILLANOS.EDU.CO	A40987267
46762330	MARIANA DIAZ RODRIGUEZ	MDIAZRODRIGUEZ@UNILLANOS.EDU.CO	M46762330
82709090	JOSE LUIS RUIZ MOLANO	JRUIZMOLANO@UNILLANOS.EDU.CO	JL82709090
82787309	PEDRO LUIS RUIZ MORENO	PRUIZMORENO@UNILLANOS.EDU.CO	PL82787309
86072892	RODRIGO MORENO DIAZ	RMORENODIAZ@UNILLANOS.EDU.CO	R86072892
827091220	ANDRES FELIPE CASTAÑO MARTINEZ	ACASTAÑO MARTINEZ@UNILLANOS.EDU.CO	AF827091220
827209090	PEDRO JOSE SANCHEZ	PSANCHEZ@UNILLANOS.EDU.CO	PJ827209090

- Cree una función para conocer el precio promedio de un elemento de protección, la función debe recibir como parámetro el código del elemento.

```
1. CREATE OR REPLACE FUNCTION precio_promedio_elemento(cod_elemento
detalle_entrada.elemento%TYPE)
2. RETURNS detalle_entrada.v_unitario%TYPE AS $$
3. DECLARE
4.     precio_promedio detalle_entrada.v_unitario%TYPE;
5. BEGIN
6.     SELECT AVG(det.v_unitario) INTO precio_promedio FROM detalle_
entrada AS det WHERE det.elemento = cod_elemento;
7.     RETURN precio_promedio;
8. END;
9. $$ LANGUAGE plpgsql;
```

	precio_promedio_elemento integer
1	150000


- Cree una función que determine si un trabajador actualmente se le han entregado todos los elementos de protección que le fueron asignados, la función debe retornar un valor entero(1 o 0).

```
1. CREATE OR REPLACE FUNCTION entregas_trabajador(id_empleado
empleados.identificacion%TYPE)
2. RETURNS INTEGER AS $$
3. DECLARE
4.     contador INTEGER;
5. BEGIN
```



```
6. SELECT COUNT(ea.Empleado) INTO contador FROM elementos_asigna
dos AS ea LEFT JOIN entrega_elementos AS ee ON ea.elemento = ee.e
lemento WHERE ee.id_entrega IS NULL AND ea.Empleado = id_Empleado
;
7. IF contador >= 1 THEN
8.     RETURN 0;
9. ELSE
10.     RETURN 1;
11. END IF;
12. END;
13. $$ LANGUAGE plpgsql;
```

```
231 SELECT entregas_trabajador(827209090);
232
```

Data Output		Messages	Notifications
			
entregas_trabajador integer			
1	1		

- Realizar una consulta que muestre: la identificación, el nombre del empleado, el cargo y una columna que indique si le entregaron (S/N) los elementos de protección.

```
1. SELECT emp.identificacion, CONCAT_WS('
',emp.nombre_1,emp.nombre_2,emp.apellido_1,emp.apellido_2) AS nom
bre_empleado,
2. c.cargo,
3. CASE
4.     WHEN entregas_trabajador(emp.identificacion) = 1
5.     THEN 'SI'
6.     ELSE 'NO'
7. END AS entrega_elementos
8. FROM empleados AS emp
9. INNER JOIN historial_laboral AS hl ON emp.identificacion = hl.emp
leado
10.     INNER JOIN cargos AS c ON hl.cargo = c.cod_cargo;
```




identificacion integer	nombre_empleado text	cargo character varying (100)	entrega_elementos text
82787309	PEDRO LUIS RUIZ MORENO	TECNICO INSTALADOR ADSL	SI
827209090	PEDRO JOSE SANCHEZ	TECNICO INSTALADOR OPTICAL	SI
827091220	ANDRES FELIPE CASTAÑO MARTINEZ	TECNICO INSTALADOR UMTS	SI
40987267	ANDREA CASTAÑO MARTINEZ	TECNICO INSTALADOR ADSL	NO
46762330	MARIANA DIAZ RODRIGUEZ	TECNICO TELMEX MEDELLIN	NO
86072892	RODRIGO MORENO DIAZ	TECNICO INSTALADOR ADSL	NO
82709090	JOSE LUIS RUIZ MOLANO	TECNICO TELMEX BOGOTA	NO
82709090	JOSE LUIS RUIZ MOLANO	TECNICO TELMEX BOGOTA	NO

- Cree una función para conocer si un estudiante está matriculado actualmente, se entiende por matriculado que el estudiante por lo menos halla inscrito una materia en un periodo y año determinado. La función debe recibir como parámetro el código del estudiante, periodo y año, debe retornar un valor booleano.

```
1. CREATE OR REPLACE FUNCTION estudiante_matriculado(cod_est
inscripciones.estudiante%TYPE, periodo_ins
inscripciones.periodo%TYPE, ano_ins INTEGER)
2. RETURNS BOOLEAN AS $$
3. DECLARE
4.     est_matriculado INTEGER;
5. BEGIN
6.     SELECT COUNT(*) INTO est_matriculado FROM inscripciones WHERE
estudiante = cod_est AND periodo = periodo_ins AND ano = ano_ins
7.     ;
8.     IF est_matriculado >= 1 THEN
9.         RETURN TRUE;
10.    ELSE
11.        RETURN FALSE;
12.    END IF;
13. END;
14. $$ LANGUAGE plpgsql;
```



```
264 SELECT estudiante_matriculado(1,1,2008);
265
266 --Cree una función que devuelva el año de
```

Data Output		Messages	Notifications
			
	estudiante_matriculado		
	boolean		
1	true		

- Cree una función que devuelva el año de la fecha que tienen el servidor. La función no recibe parámetros, pero devuelve un valor numérico.

```
1. CREATE OR REPLACE FUNCTION year_servidor()
2. RETURNS INTEGER AS $$
3. BEGIN
4.     RETURN EXTRACT(YEAR FROM NOW());
5. END;
6. $$ LANGUAGE plpgsql;
7. SELECT year_servidor();
```

	year_servidor	
	integer	
1	2023	

- Cree una función que determine el semestre del año en que se encuentra actualmente, tener como base la fecha del servidor. La función no recibe parámetros, pero devuelve un valor numérico.


```
1. CREATE OR REPLACE FUNCTION semestre_servidor()
2. RETURNS INTEGER AS $$
3. BEGIN
4.     IF EXTRACT(MONTH FROM NOW()) <= 6 THEN
5.         RETURN 1;
6.     ELSE
7.         RETURN 2;
8.     END IF;
9. END;
10. $$ LANGUAGE plpgsql;
```



	semestre_servidor
	integer
1	1

- Cree una función para conocer el promedio de carrera de un estudiante, solo tener en cuenta las materias aprobadas, la función debe recibir como parámetro el código del estudiante.

```
1. CREATE OR REPLACE FUNCTION promedio_carrera(cod_est
   inscripciones.estudiante%TYPE)
2. RETURNS inscripciones.nota%TYPE AS $$
3. DECLARE
4.     promedio inscripciones.nota%TYPE;
5. BEGIN
6.     SELECT AVG(ins.nota) INTO promedio FROM inscripciones AS ins
   WHERE ins.estudiante = cod_est AND ins.nota >=3;
7.     RETURN promedio;
8. END;
9. $$ LANGUAGE plpgsql;
```

301	SELECT promedio_carrera(1);
302	
Data Output Messages Notifications	
	
	promedio_carrera
	numeric
1	4.37

- Cree un procedimiento que muestre en pantalla la siguiente información de un estudiante:



- )( Código.
- )( Facultad.
- )( Programa.
- )( Nombres y apellidos.
- Promedio de carrera de materias aprobadas.
- )( Estudiante está matriculado actualmente(S/N).
- )( Año actual.
- )( Período actual.

Además, esta información debe quedar almacenada en la siguiente tabla:

```
CREATE TABLE TMP_ESTUDIANTES
(
  CODIGO          INTEGER NOT NULL,
  FACULTAD        VARCHAR2(50) NOT NULL,
  PROGRAMA        VARCHAR2(50) NOT NULL,
  ESTUDIANTE      VARCHAR2(200) NOT NULL,
  PROMEDIO        INTEGER (5,3) NOT NULL,
  MATRICULADO     VARCHAR2(1)  NOT NULL,
  AÑO             INTEGER      NOT NULL,
  PERIODO         INTEGER      NOT NULL
);
```

Pero, solo para aquellos estudiantes cuyo promedio de carrera sea mayor o igual a 3.8. Cada vez que se llame el procedimiento, este debe vaciar la tabla TMP\_ESTUDIANTES.

```
1. CREATE OR REPLACE PROCEDURE informacion_estudiantes ()
2. AS $$
3. DECLARE
4.   cur_estudiantes CURSOR FOR
5.   SELECT est.codigo AS codigo_estudiante,
6.   fac.nombre AS facultad, pr.nombre AS programa,
7.   CONCAT_WS('
8.   ',est.nombres,est.apellido1,est.apellido2) AS nombre_completo FROM
9.   M estudiante AS est
10.  INNER JOIN programa AS pr ON est.facultad = pr.facultad AND e
11.  st.programa = pr.id_programa
12.  INNER JOIN facultad AS fac ON est.facultad = fac.id_facultad;
13.
14.  prom_est tmp_estudiantes.promedio%TYPE;
15.  est_matriculado VARCHAR(1);
16.  ano_actual INTEGER;
17.  periodo_actual INTEGER;
18. BEGIN
19.   DELETE FROM tmp_estudiantes;
20.   ano_actual := year_servidor();
21.   periodo_actual := semestre_servidor();
```



```
18.      RAISE NOTICE 'informacion academica de los
estudiantes';
19.      FOR v_estudiante IN cur_estudiantes LOOP
20.          prom_est
:= promedio_carrera(v_estudiante.codigo_estudiante);
21.          IF estudiante_matriculado(v_estudiante.codigo_estud
iante,periodo_actual,ano_actual) = TRUE THEN
22.              est_matriculado := 'S';
23.          ELSE
24.              est_matriculado := 'N';
25.          END IF;
26.          IF prom_est >=3.8 THEN
27.              RAISE NOTICE 'CODIGO--->
%',v_estudiante.codigo_estudiante;
28.              RAISE NOTICE 'FACULTAD--->
%',v_estudiante.facultad;
29.              RAISE NOTICE 'PROGRAMA--->
%',v_estudiante.programa;
30.              RAISE NOTICE 'NOMBRES Y APELLIDOS--->
%',v_estudiante.nombre_completo;
31.              RAISE NOTICE 'PROMEDIO DE CARRERA--->
%',prom_est;
32.              RAISE NOTICE 'MATRICULADO--->
%',est_matriculado;
33.              RAISE NOTICE 'AÑO ACTUAL---> %',ano_actual;
34.              RAISE NOTICE 'PERIODO ACTUAL--->
%',periodo_actual;
35.              INSERT INTO tmp_estudiantes(codigo,facultad,pro
grama,estudiante,promedio,matriculado,ano,periodo)
36.              VALUES(v_estudiante.codigo_estudiante,v_estudia
nte.facultad,v_estudiante.programa,v_estudiante.nombre_completo
37.              ,prom_est,est_matriculado,ano_actual,periodo_ac
tual);
38.          END IF;
39.      END LOOP;
40.      COMMIT;
41.  END;
42.  $$ LANGUAGE plpgsql;
43.  CALL informacion_estudiantes();
44.  SELECT * FROM tmp_estudiantes;
```





ig (50)	programa character varying (50)	estudiante character varying (200)	promedio numeric (5,3)	matriculado character vai
ENIERIAS	INGENIERIA DE SISTEMAS	juan calderon martinez	4.370	N
ENIERIAS	INGENIERIA DE SISTEMAS	carlos moreno	3.900	N
ENIERIAS	INGENIERIA DE SISTEMAS	pedro ruiz moreno	4.800	N
ENIERIAS	INGENIERIA DE SISTEMAS	sandra morales ruiz	3.830	N
ENIERIAS	INGENIERIA INDUSTRIAL	milena castaño	4.400	N
ENIERIAS	INGENIERIA INDUSTRIAL	jose martinez	3.870	N
	LICENCIATURA EN MATEMATICAS Y FISICA	luz arias	4.600	N
	LICENCIATURA EN MATEMATICAS Y FISICA		4.670	N

- Cree una función que retorne el número de estudiantes inscritos por materia, la función debe recibir como parámetros el código de la materia, el código del programa, código de facultad, el año y periodo.

```
1. CREATE OR REPLACE FUNCTION est_inscritos_materia(cod_mat
materias.id_materia%TYPE, cod_pro programa.id_programa%TYPE,
cod_fac facultad.id_facultad%TYPE, ano_ins
inscripciones.ano%TYPE, periodo_ins inscripciones.periodo%TYPE)
2. RETURNS INTEGER AS $$
3. DECLARE
4.     cantidad INTEGER;
5. BEGIN
6.     SELECT COUNT(*) INTO cantidad FROM inscripciones AS ins WHERE
ins.materia = cod_mat AND
7.     ins.programa = cod_pro AND ins.facultad = cod_fac AND ins.ano
= ano_ins AND ins.periodo = periodo_ins;
8.     RETURN cantidad;
9. END;
10. $$ LANGUAGE plpgsql;
11.
12. SELECT est_inscritos_materia(2,1,1,2008,1);
```

est_inscritos_materia	integer
1	5

- Cree un dato compuesto de tipo tabla, que almacene la información del punto 18 (Solo la primera parte), con base en esta información debe mostrar:
  1. El índice de la tabla y los datos.
  2. Total de datos de la tabla.



3. Validar si la clave 10 existe, y si existe debe mostrar la información que contiene, si no, debe indicar que no existe.
4. Mostrar la información de la primera y la última clave.
5. Eliminar los índices de la tabla comprendidos entre 1 y 5, utilice la función delete (m, n).

```
1. CREATE OR REPLACE PROCEDURE informacion_18()  
2. AS $$  
3. DECLARE  
4.     inscritos_tabla inscripciones_tabla[];  
5.     rm_inscritos_tabla inscripciones_tabla[];  
6.     pos INTEGER := 0;  
7.     cur_inscripciones CURSOR FOR  
8.     SELECT ROW_NUMBER() OVER() AS indice, t.materia, t.cantidad  
FROM(SELECT nombre AS materia, COUNT(id_materia) AS cantidad FROM  
    inscripciones ins  
9.     INNER JOIN materias ON ins.materia = materias.id_materia  
10.    GROUP BY nombre) AS t;  
11.  
12. BEGIN  
13.     RAISE  
14.     NOTICE '/////////////////////////////////////////';  
15.     RAISE NOTICE '1. EL INDICE DE LA TABLA Y LOS  
16.     DATOS:';  
17.     FOR v_inscritos IN cur_inscripciones LOOP  
18.         RAISE NOTICE '-----'  
19.         inscritos_tabla[v_inscritos.indice]:= v_inscritos;  
20.         RAISE NOTICE 'INDICE:'  
21.         '%',inscritos_tabla[v_inscritos.indice].indice;  
22.         RAISE NOTICE 'MATERIA:'  
23.         '%',inscritos_tabla[v_inscritos.indice].materia;  
24.         RAISE NOTICE 'CANTIDAD:'  
25.         '%',inscritos_tabla[v_inscritos.indice].cantidad;  
26.     END LOOP;  
27.     RAISE  
28.     NOTICE '/////////////////////////////////////////';  
29.     RAISE NOTICE '2. TOTAL DE DATOS DE LA TABLA.';  
30.     RAISE NOTICE '-----'  
31.     RAISE NOTICE 'TOTAL DE DATOS:'  
32.     '%',ARRAY_LENGTH(inscritos_tabla,1);
```



```
26.          RAISE
NOTICE '/////////////////////////////////////////';
27.          RAISE NOTICE '3. VALIDAR SI LA CLAVE 10 EXISTE, Y
SI EXISTE DEBE MOSTRAR LA INFORMACION QUE CONTIENE, SI NO, DEBE
INDICAR QUE NO EXISTE.';
28.          IF inscritos_tabla[10] IS NOT NULL THEN
29.              RAISE NOTICE '-----
-----';
30.              RAISE NOTICE 'INDICE:
%',inscritos_tabla[10].indice;
31.              RAISE NOTICE 'MATERIA:
%',inscritos_tabla[10].materia;
32.              RAISE NOTICE 'CANTIDAD:
%',inscritos_tabla[10].cantidad;
33.          ELSE
34.              RAISE NOTICE '-----
-----';
35.              RAISE NOTICE 'LA CLAVE 10 NO EXISTE';
36.          END IF;
37.          RAISE
NOTICE '/////////////////////////////////////////';
38.          RAISE NOTICE '4. MOSTRAR LA INFORMACION DE LA
PRIMERA Y LA ULTIMA CLAVE';
39.          RAISE NOTICE '-----
-----';
40.          RAISE NOTICE 'PRIMER INDICE:
%',inscritos_tabla[ARRAY_LOWER(inscritos_tabla,1)].indice;
41.          RAISE NOTICE 'MATERIA:
%',inscritos_tabla[ARRAY_LOWER(inscritos_tabla,1)].materia;
42.          RAISE NOTICE 'CANTIDAD:
%',inscritos_tabla[ARRAY_LOWER(inscritos_tabla,1)].cantidad;
43.          RAISE NOTICE '-----
-----';
44.          RAISE NOTICE 'ULTIMO INDICE:
%',inscritos_tabla[ARRAY_UPPER(inscritos_tabla,1)].indice;
45.          RAISE NOTICE 'MATERIA:
%',inscritos_tabla[ARRAY_UPPER(inscritos_tabla,1)].materia;
46.          RAISE NOTICE 'CANTIDAD:
%',inscritos_tabla[ARRAY_UPPER(inscritos_tabla,1)].cantidad;
47.          RAISE
NOTICE '/////////////////////////////////////////';
48.          RAISE NOTICE '5. ELIMINAR LOS INDICES DE LA TABLA
COMPRENDIDOS ENTRE 1 Y 5, UTILICE LA FUNCION DELETE(M,N)';
49.          RAISE NOTICE '-----
-----';
```



```
50.         FOR i IN 1..ARRAY_LENGTH(inscritos_tabla, 1) LOOP
51.             IF i > 5 THEN
52.                 pos := pos+1;
53.                 rm_inscritos_tabla[pos] := inscrito
s_tabla[i];
54.             END IF;
55.         END LOOP;
56.         RAISE NOTICE 'ELIMINADOS';
57.         RAISE
NOTICE '/////////////////////////////////////////';
58.         RAISE NOTICE 'DATOS ACTUALES';
59.         RAISE NOTICE '-----'
-----';
60.         inscritos_tabla := rm_inscritos_tabla;
61.         FOR i IN 1..ARRAY_LENGTH(inscritos_tabla,1) LOOP
62.             RAISE NOTICE 'INDICE:
%',inscritos_tabla[i].indice;
63.             RAISE NOTICE 'MATERIA:
%',inscritos_tabla[i].materia;
64.             RAISE NOTICE 'CANTIDAD:
%',inscritos_tabla[i].cantidad;
65.             RAISE NOTICE '-----'
-----';
66.         END LOOP;
67.     END;
68.    $$ LANGUAGE plpgsql;
69.    CALL informacion_18();
```



```
NOTICE: //////////////////////////////////////
NOTICE: 1. EL INDICE DE LA TABLA Y LOS DATOS:
NOTICE: -----
NOTICE: INDICE: 1
NOTICE: MATERIA: INTRODUCCION A LA GEOMETRIA
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 2
NOTICE: MATERIA: INFORMATICA BASICA
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 3
NOTICE: MATERIA: MATEMATICAS I
NOTICE: CANTIDAD: 26
NOTICE: -----
NOTICE: INDICE: 4
NOTICE: MATERIA: FISICA I
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 5
NOTICE: MATERIA: PROGRAMACION I
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 6
NOTICE: MATERIA: INFORMATICA AVANZADA
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 7
NOTICE: MATERIA: ALGEBRA LINEAL
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 8
NOTICE: MATERIA: ETICA
NOTICE: CANTIDAD: 13
NOTICE: //////////////////////////////////////
NOTICE: 2. TOTAL DE DATOS DE LA TABLA.
NOTICE: -----
NOTICE: TOTAL DE DATOS: 8
NOTICE: //////////////////////////////////////
NOTICE: 3. VALIDAR SI LA CLAVE 10 EXISTE, Y SI EXISTE DEBE MOSTRAR LA INFORMACION QUE
CONTIENE, SI NO, DEBE INDICAR QUE NO EXISTE.
NOTICE: -----
NOTICE: LA CLAVE 10 NO EXISTE
NOTICE: //////////////////////////////////////
```



```
NOTICE: 4. MOSTRAR LA INFORMACION DE LA PRIMERA Y LA ULTIMA CLAVE
NOTICE: -----
NOTICE: PRIMER INDICE: 1
NOTICE: MATERIA: INTRODUCCION A LA GEOMETRIA
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: ULTIMO INDICE: 8
NOTICE: MATERIA: ETICA
NOTICE: CANTIDAD: 13
NOTICE: //////////////////////////////////////
NOTICE: 5. ELIMINAR LOS INDICES DE LA TABLA COMPRENDIDOS ENTRE 1 Y 5, UTILICE LA FUNCION
DELETE(M,N)
NOTICE: -----
NOTICE: ELIMINADOS
NOTICE: //////////////////////////////////////
NOTICE: DATOS ACTUALES
NOTICE: -----
NOTICE: INDICE: 6
NOTICE: MATERIA: INFORMATICA AVANZADA
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 7
NOTICE: MATERIA: ALGEBRA LINEAL
NOTICE: CANTIDAD: 13
NOTICE: -----
NOTICE: INDICE: 8
NOTICE: MATERIA: ETICA
NOTICE: CANTIDAD: 13
NOTICE: -----
CALL
```

Query returned successfully in 115 msec.