

# 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;
```





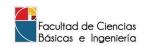
#### 10. \$\$ LANGUAGE PLPGSQL;

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

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

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

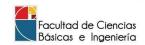
```
1. DO $$
2. DECLARE
3. numerol INTEGER := 7;
4. numero2 INTEGER := 6;
5. BEGIN
6. IF numero1 > numero2 THEN
          RAISE NOTICE '% ES MAYOR QUE %', numero1, numero2;
7.
8.
    ELSIF numero1 = numero2 THEN
9.
       RAISE NOTICE '% ES IGUAL QUE %', numero1, numero2;
10.
           ELSE
11.
   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

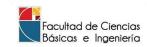
\* Nota: El campo ID debe ser un número primo y el campo descripción debe tener la siguiente cadena de texto: 'El identificador es 2', donde el numero es el id que generarón.

CREATE TABLE

Query returned successfully in 66 msec.

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





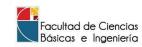
id [PK] integer	descripcion character varying (30)
2	El identificador es 2
3	El identificador es 3
5	El identificador es 5
7	El identificador es 7
11	El identificador es 11
13	El identificador es 13
17	El identificador es 17
19	El identificador es 19

 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 duración.

nero aracter 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
	ki .	0000 11 14	00700000	0000 10 15	20

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





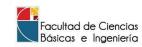
```
1. CREATE OR REPLACE PROCEDURE pares()
2. AS $$
3. BEGIN
4.
           FOR numero IN 0..100 LOOP
5.
                    IF MOD (numero, 2) = 0 THEN
6.
                            RAISE NOTICE 'NUMERO: %', numero;
7.
                    END IF;
8.
           END LOOP;
9. END;
10.
        $$ LANGUAGE 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
                   primo := FALSE;
9.
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 (numerol INTEGER, nume
  ro2 INTEGER)
2. AS $$
3. BEGIN
4.
     IF numero1 > numero2 THEN
5.
           RAISE NOTICE '% ES MAYOR QUE %', numero1, numero2;
6.
      ELSIF numero1 = numero2 THEN
7.
          RAISE NOTICE '% ES IGUAL QUE %', numero1, numero2;
      ELSE
9.
           RAISE NOTICE '% ES MAYOR QUE %', numero2, numero1;
10.
            END IF;
11.
        END;
12.
        $$ LANGUAGE plpqsql;
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 numero al final del nombre de la dirección de correo, el número es un consecutivo que debe iniciar en 1.

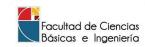
Función para buscar correos repetidos:

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

Procedimiento para ingresar la nueva dirección de correo:

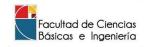
```
1. CREATE OR REPLACE PROCEDURE actualizar_correos()
2. AS $$
3. DECLARE
```





```
cur empleados CURSOR FOR SELECT nombre 1, nombre 2,
 apellido 1, apellido 2, identificacion FROM empleados;
5. correo nuevo empleados.email%TYPE;
      clave nueva empleados.clave%TYPE;
6.
      dominio VARCHAR (17) := '@UNILLANOS.EDU.CO';
      contador INTEGER;
9. BEGIN
10.
            -- Creamos un cursor para recorrer la tabla de
empleados
            FOR empleado IN cur empleados LOOP
11.
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);
                --Validamos que no se repita la direccion de correo
16.
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
               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();
```

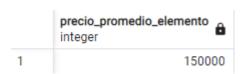




ficacion nteger	nombre_completo text	email character varying (100)	clave character varying
40987267	ANDREA CASTA♦O MARTINEZ	ACASTA♦OMARTINEZ1@UNILLANOS.EDU.CO	A40987267
46762330	MARIANA DIAZ RODRIGUEZ	MDIAZRODRIGUEZ@UNILLANOS.EDU.CO	M46762330
82709090	JOSE LUIS RUIZ MOLANO	JRUIZMOLANO@UNILLANOS.EDU.CO	JL82709090
82787309	PEDRO LUIS RUIZ MORENO	PRUIZMORENO@UNILLANOS.EDU.CO	PL82787309
86072892	RODRIGO MORENO DIAZ	RMORENODIAZ@UNILLANOS.EDU.CO	R86072892
827091220	ANDRES FELIPE CASTA O MARTINEZ	ACASTA♦OMARTINEZ@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.

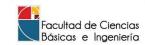
```
    CREATE OR REPLACE FUNCTION precio_promedio_elemento (cod_elemento detalle_entrada.elemento%TYPE)
    RETURNS detalle_entrada.v_unitario%TYPE AS $$
    DECLARE
    precio_promedio detalle_entrada.v_unitario%TYPE;
    BEGIN
    SELECT AVG(det.v_unitario) INTO precio_promedio FROM detalle_entrada AS det WHERE det.elemento = cod_elemento;
    RETURN precio_promedio;
    END;
    $$ LANGUAGE plpgsql;
```



 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).

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



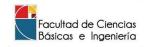


```
SELECT COUNT (ea.empleado) INTO contador FROM elementos asigna
   dos AS ea LEFT JOIN entrega elementos AS ee ON ea.elemento = ee.e
   lemento WHERE ee.id entrega IS NULL AND ea.empleado = id empleado
7.
       IF contador >= 1 THEN
8.
           RETURN 0;
9.
       ELSE
10.
                 RETURN 1;
11.
             END IF;
12.
        END;
13.
         $$ LANGUAGE plpgsql;
                  SELECT entregas_trabajador(827209090);
             232
             Data Output
                         Messages
                                    Notifications
             =<sub>+</sub>
                  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
           WHEN entregas trabajador(emp.identificacion) = 1
4.
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;
```



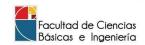


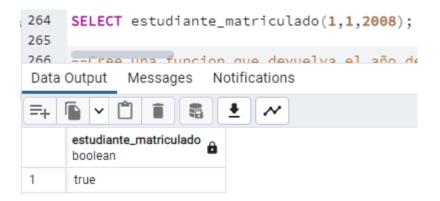
identificacion integer	nombre_empleado text	cargo character varying (100) €	entrega_elementos text
82787309	PEDRO LUIS RUIZ MORENO	TECNICO INSTALADOR ADSL	SI
827209090	PEDRO JOSE SANCHEZ	TECNICO INSTALADOR OPTICAL	SI
827091220	ANDRES FELIPE CASTA O MARTINEZ	TECNICO INSTALADOR UMTS	SI
40987267	ANDREA CASTA♦O 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
00707000	DEDDO LUIO DUIZ MODENO	INODEOTOD TELEFONIA DUDLIOA	01

 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 $$
3. DECLARE
4.
          est matriculado INTEGER;
5. BEGIN
      SELECT COUNT(*) INTO est matriculado FROM inscripciones WHERE
   estudiante = cod est AND periodo = periodo ins AND ano = ano ins
7.
8.
      IF est matriculado >= 1 THEN
9.
         RETURN TRUE;
           ELSE
10.
11.
                RETURN FALSE;
12.
            END IF;
13.
       END;
14.
      $$ LANGUAGE plpgsql;
```







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

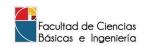
```
1. CREATE OR REPLACE FUNCTION year_servidor()
2. RETURNS INTEGER AS $$
3. BEGIN
4. RETURN EXTRACT(YEAR FROM NOW());
5. END;
6. $$ LANGUAGE plpgsql;
7. SELECT year_servidor();
```



 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;
```

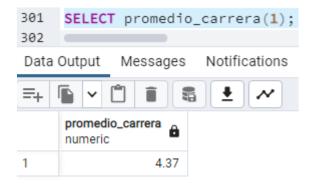






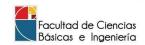
 Cree una función para conocer el promedio de carrera de un estudiante, solo tener en cuenta las materias aprobadas, la función debe recibir como parámetro el código del estudiante.

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



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





```
H Código.

HH Facultad.

HH Programa.

H ◆ Promedio de carrera de materias aprobadas.

The Promedio de carrera de materias aprobadas.
```

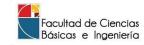
Además, esta información debe quedar almacena en la siguiente tabla:

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

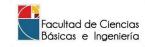
```
1. CREATE OR REPLACE PROCEDURE informacion estudiantes()
2. AS $$
3. DECLARE
       cur estudiantes CURSOR FOR
4.
       SELECT est.codigo AS codigo estudiante,
  fac.nombre AS facultad, pr.nombre AS programa,
       CONCAT WS ( '
   ',est.nombres,est.apellido1,est.apellido2) AS nombre completo FRO
  M estudiante AS est
       INNER JOIN programa AS pr ON est.facultad = pr.facultad AND e
7.
  st.programa = pr.id programa
8.
       INNER JOIN facultad AS fac ON est.facultad = fac.id facultad;
9.
10.
            prom est tmp estudiantes.promedio%TYPE;
11.
            est matriculado VARCHAR(1);
12.
            ano actual INTEGER;
13.
            periodo actual INTEGER;
14.
        BEGIN
15.
            DELETE FROM tmp estudiantes;
16.
            ano actual := year servidor();
17.
            periodo actual := semestre servidor();
```





```
RAISE NOTICE 'informacion academica de los
 estudiantes';
           FOR v estudiante IN cur estudiantes LOOP
19.
20.
                prom est
  := promedio carrera(v estudiante.codigo estudiante);
                IF estudiante matriculado (v estudiante.codigo estud
21.
  iante,periodo actual, ano actual) = TRUE THEN
                    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;
                    RAISE NOTICE 'FACULTAD--->
  %',v estudiante.facultad;
29.
                    RAISE NOTICE 'PROGRAMA--->
  %',v estudiante.programa;
                    RAISE NOTICE 'NOMBRES Y APELLIDOS--->
  %',v estudiante.nombre completo;
                    RAISE NOTICE 'PROMEDIO DE CARRERA--->
31.
  %',prom est;
32.
                    RAISE NOTICE 'MATRICULADO--->
  %',est matriculado;
33.
                    RAISE NOTICE 'AÑO ACTUAL---> %', ano actual;
                    RAISE NOTICE 'PERIODO ACTUAL--->
34.
  %',periodo actual;
                     INSERT INTO tmp estudiantes(codigo, facultad, pro
35.
  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();
        SELECT * FROM tmp estudiantes;
44.
```





ıg (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
	LIGENOLATURA EN MATEMATICA O VICIOIOA		4.070	k1

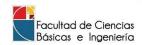
 Cree una función que retorne el número de estudiantes inscritos por materia, la función debe recibir como parámetros el código de la materia, el código del programa, código de facultad, el año y periodo.

```
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 $$
3. DECLARE
          cantidad INTEGER;
5. BEGIN
      SELECT COUNT(*) INTO cantidad FROM inscripciones AS ins WHERE
   ins.materia = cod mat AND
      ins.programa = cod pro AND ins.facultad = cod fac AND ins.ano
   = ano ins AND ins.periodo = periodo ins;
8.
      RETURN cantidad;
9. END;
10.
        $$ LANGUAGE plpgsql;
11.
        SELECT est inscritos materia(2,1,1,2008,1);
12.
```

	est_inscritos_materia integer	â
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).

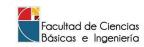
```
1. CREATE OR REPLACE PROCEDURE informacion 18()
2. AS $$
3. DECLARE
4. inscritos tabla inscripciones tabla[];
        rm inscritos tabla inscripciones tabla[];
6.
        pos INTEGER := 0;
    cur inscripciones CURSOR FOR
   SELECT ROW NUMBER() OVER() AS indice, t.materia, t.cantidad
  FROM (SELECT 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
 NOTICE '/////////////;
14.
             RAISE NOTICE '1. EL INDICE DE LA TABLA Y LOS
 DATOS: ';
            FOR v inscritos IN cur inscripciones LOOP
15.
16.
                    RAISE NOTICE '-----
            -----;
17.
                     inscritos tabla[v inscritos.indice]:= v ins
  critos;
             RAISE NOTICE 'INDICE:
  %', inscritos_tabla[v_inscritos.indice].indice;
                    RAISE NOTICE 'MATERIA:
  %', inscritos tabla[v inscritos.indice].materia;
                     RAISE NOTICE 'CANTIDAD:
20.
  %', inscritos tabla[v inscritos.indice].cantidad;
21.
             END LOOP;
              RAISE
 23.
             RAISE NOTICE '2. TOTAL DE DATOS DE LA TABLA.';
              RAISE NOTICE '----
24.
25.
                     RAISE NOTICE 'TOTAL DE DATOS:
%',ARRAY LENGTH(inscritos tabla,1);
```





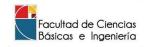
```
26.
           RAISE
27. RAISE NOTICE '3. VALIDAR SI LA CLAVE 10 EXISTE, Y
 SI EXISTE DEBE MOSTRAR LA INFORMACION QUE CONTIENE, SI NO, DEBE
 INDICAR OUE NO EXISTE.';
28.
       IF inscritos tabla[10] IS NOT NULL THEN
                  RAISE NOTICE '-----
               -----;
                  RAISE NOTICE 'INDICE:
%',inscritos tabla[10].indice;
                  RAISE NOTICE 'MATERIA:
%',inscritos tabla[10].materia;
                  RAISE NOTICE 'CANTIDAD:
%',inscritos tabla[10].cantidad;
           ELSE
33.
34.
                  RAISE NOTICE '----
       ----';
                  RAISE NOTICE 'LA CLAVE 10 NO EXISTE';
36.
           END IF;
37.
            RAISE
RAISE NOTICE '4. MOSTRAR LA INFORMACION DE LA
PRIMERA Y LA ULTIMA CLAVE';
39.
           RAISE NOTICE '----
         -----';
      RAISE NOTICE 'PRIMER INDICE:
%', inscritos tabla [ARRAY LOWER (inscritos tabla, 1)].indice;
    RAISE NOTICE 'MATERIA:
 %',inscritos tabla[ARRAY LOWER(inscritos tabla,1)].materia;
   RAISE NOTICE 'CANTIDAD:
 %',inscritos tabla[ARRAY LOWER(inscritos tabla,1)].cantidad;
            RAISE NOTICE '----
43.
        ----';
           RAISE NOTICE 'ULTIMO INDICE:
 %',inscritos tabla[ARRAY UPPER(inscritos tabla,1)].indice;
45. RAISE NOTICE 'MATERIA:
%', inscritos tabla [ARRAY UPPER(inscritos tabla, 1)].materia;
            RAISE NOTICE 'CANTIDAD:
 %',inscritos tabla[ARRAY UPPER(inscritos tabla,1)].cantidad;
            RAISE
 48. RAISE NOTICE '5. ELIMINAR LOS INDICES DE LA TABLA
COMPRENDIDOS ENTRE 1 Y 5, UTILICE LA FUNCION DELETE (M, N) ';
           RAISE NOTICE '----
    -----';
```





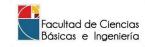
```
50.
               FOR i IN 1..ARRAY LENGTH (inscritos tabla, 1) LOOP
51.
                      IF i > 5 THEN
52.
                             pos := pos+1;
53.
                             rm inscritos tabla[pos] := inscrito
s tabla[i];
54.
                      END IF;
55.
             END LOOP;
56.
              RAISE NOTICE 'ELIMINADOS';
              RAISE
RAISE NOTICE 'DATOS ACTUALES';
58.
              RAISE NOTICE '-----
          -----';
60.
              inscritos tabla := rm inscritos tabla;
61.
              FOR i IN 1..ARRAY LENGTH (inscritos tabla, 1) LOOP
                      RAISE NOTICE 'INDICE:
  %',inscritos tabla[i].indice;
                     RAISE NOTICE 'MATERIA:
  %',inscritos tabla[i].materia;
                     RAISE NOTICE 'CANTIDAD:
  %',inscritos tabla[i].cantidad;
                     RAISE NOTICE '-----
65.
66.
              END LOOP;
67. END;
68. $$ LANGUAGE plpgsql;
69. CALL informacion_18();
```





	//////////////////////////////////////
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:	///////////////////////////////////////
	3. VALIDAR SI LA CLAVE 10 EXISTE, Y SI EXISTE DEBE MOSTRAR LA INFORMACION QUE
	, SI NO, DEBE INDICAR QUE NO EXISTE.
	´
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: 5. ELIMINAR LOS INDICES DE LA TABLA COMPRENDIDOS ENTRE 1 Y 5, UTILICE LA FUNCION DELETE(M,N) NOTICE: -----NOTICE: ELIMINADOS 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.