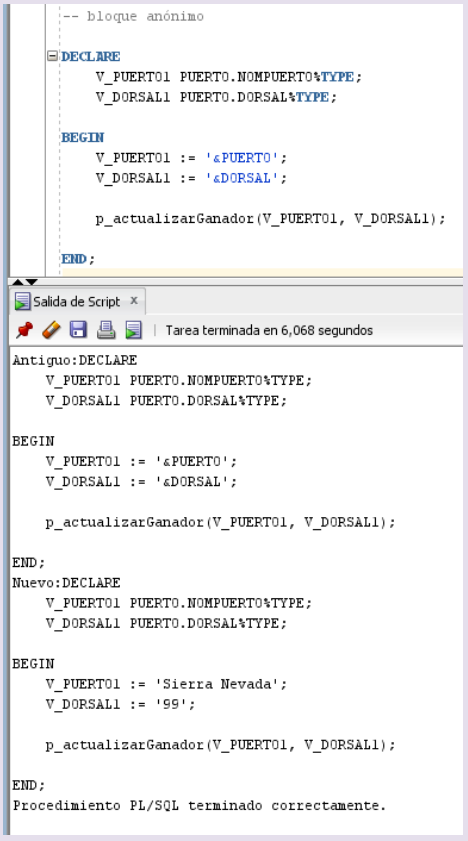
PL/SQL - Ejercicios de refuerzo

Nota previa: Todos los scripts deben ir acompañados de un bloque anónimo y del correspondiente tratamiento de excepciones.

1. Crea un procedimiento que actualice el ganador del premio de un puerto.

```
CREATE OR REPLACE PROCEDURE p_actualizarGanador(V_PUERTO IN PUERTO.NOMPUERTO%TYPE,
V_DORSAL IN PUERTO.DORSAL%TYPE)

IS
   NO_EXISTE_PUERTO EXCEPTION;

BEGIN
   UPDATE PUERTO
   SET DORSAL = V_DORSAL
   WHERE NOMPUERTO = V_PUERTO;

   IF SQL%ROWCOUNT = 0 THEN
      RAISE NO_EXISTE_PUERTO;
   END IF;

EXCEPTION
   WHEN NO_EXISTE_PUERTO THEN
      DBMS_OUTPUT.PUT_LINE('No existe ese puerto');

END p_actualizarGanador;
```


```
-- bloque anónimo

DECLARE
   V_PUERTO1 PUERTO.NOMPUERTO%TYPE;
   V_DORSAL1 PUERTO.DORSAL%TYPE;

BEGIN
   V_PUERTO1 := '&PUERTO';
   V_DORSAL1 := '&DORSAL';

   p_actualizarGanador(V_PUERTO1, V_DORSAL1);

END;
```

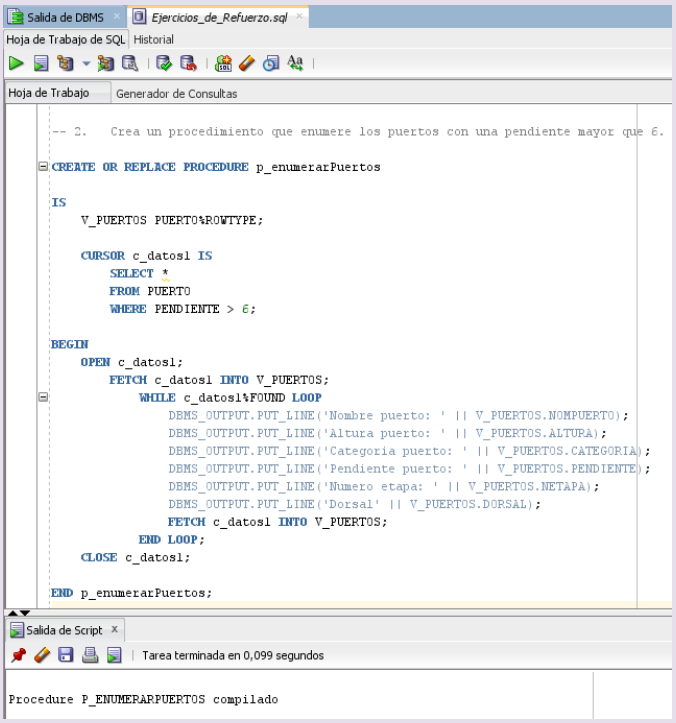


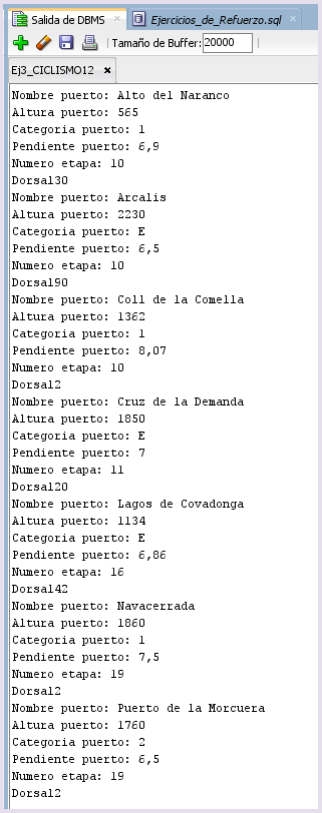

```
-- bloque anónimo

DECLARE
   V_PUERTO1 PUERTO.NOMPUERTO%TYPE;
   V_DORSAL1 PUERTO.DORSAL%TYPE;

BEGIN
   V_PUERTO1 := '&PUERTO';
   V_DORSAL1 := '&DORSAL';

   p_actualizarGanador(V_PUERTO1, V_DORSAL1);

END;
```

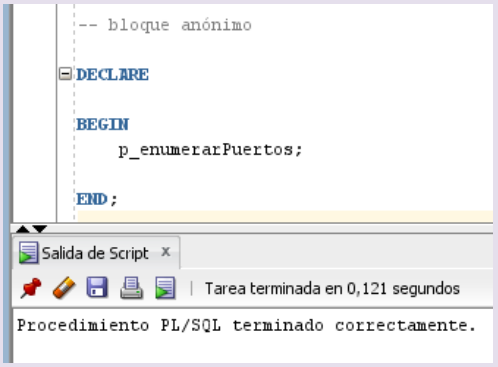
	NOMPUERTO	ALTURA	CATEGORIA	PENDIENTE	NETAPA	DORSAL
1	Alto del Naranco	565	1	6,9	10	30
2	Arcalis	2230	E	6,5	10	90
3	Cerler-Circo de Ampriu	2500	E	5,87	11	9
4	Coll de la Comella	1362	1	8,07	10	2
5	Coll de Ordino	1980	E	5,3	10	7
6	Cruz de la Demanda	1850	E	7	11	20
7	Lagos de Covadonga	1134	E	6,86	16	42
8	Navacerrada	1860	1	7,5	19	2
9	Puerto de Alisas	672	1	5,8	15	1
10	Puerto de la Morcuera	1760	2	6,5	19	2
11	Puerto de Mijares	1525	1	4,9	18	24
12	Puerto de Navalmoral	1521	2	4,3	18	2
13	Puerto de Pedro Bernardo	1250	1	4,2	18	25
14	Sierra Nevada	2500	E	8	2	99

2. Crea un procedimiento que enumere los puertos con una pendiente mayor que 6. Utiliza ambos tipos de cursor explícito.

```
CREATE OR REPLACE PROCEDURE p_enumerarPuertos

IS
   V_PUERTOS PUERTO%ROWTYPE;

   CURSOR c_datos1 IS
      SELECT *
      FROM PUERTO
      WHERE PENDIENTE > 6;

BEGIN
   OPEN c_datos1;
      FETCH c_datos1 INTO V_PUERTOS;
         WHILE c_datos1%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE('Nombre puerto: ' ||
V_PUERTOS.NOMPUERTO);
            DBMS_OUTPUT.PUT_LINE('Altura puerto: ' ||
V_PUERTOS.ALTURA);
            DBMS_OUTPUT.PUT_LINE('Categoria puerto: ' ||
V_PUERTOS.CATEGORIA);
            DBMS_OUTPUT.PUT_LINE('Pendiente puerto: ' ||
V_PUERTOS.PENDIENTE);
            DBMS_OUTPUT.PUT_LINE('Numero etapa: ' ||
V_PUERTOS.NETAPA);
            DBMS_OUTPUT.PUT_LINE('Dorsal' ||
V_PUERTOS.DORSAL);
            FETCH c_datos1 INTO V_PUERTOS;
         END LOOP;
   CLOSE c_datos1;

END p_enumerarPuertos;

-- bloque anónimo

DECLARE

BEGIN
   p_enumerarPuertos;

END;
```

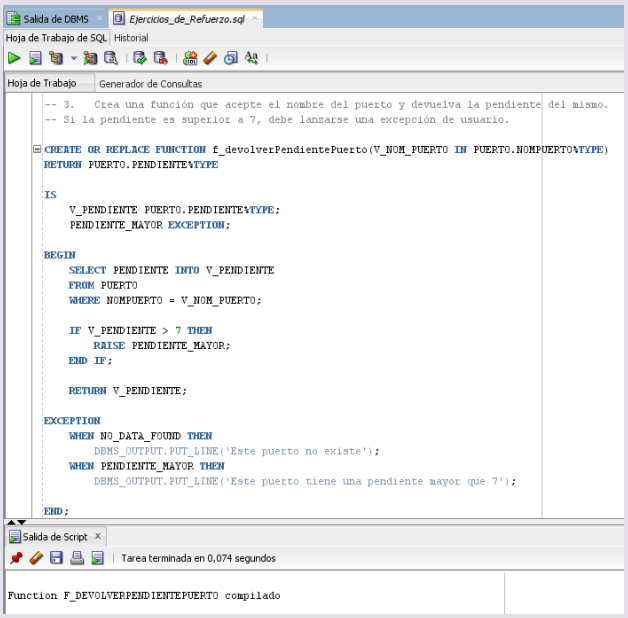




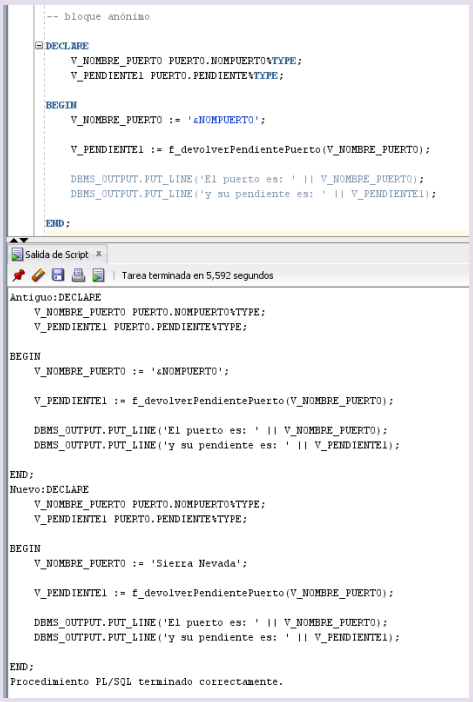


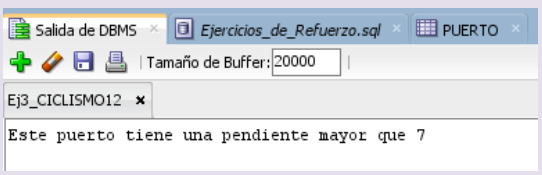
3. Crea una función que acepte el nombre del puerto y devuelva la pendiente del mismo. Si la pendiente es superior a 7, debe lanzarse una excepción de usuario.

```
CREATE OR REPLACE FUNCTION f_devolverPendientePuerto(V_NOM_PUERTO IN
PUERTO.NOMPUERTO%TYPE)
RETURN PUERTO.PENDIENTE%TYPE

IS
   V_PENDIENTE PUERTO.PENDIENTE%TYPE;
   PENDIENTE_MAYOR EXCEPTION;

BEGIN
   SELECT PENDIENTE INTO V_PENDIENTE
   FROM PUERTO
   WHERE NOMPUERTO = V_NOM_PUERTO;

   IF V_PENDIENTE > 7 THEN
      RAISE PENDIENTE_MAYOR;
   END IF;

   RETURN V_PENDIENTE;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Este puerto no existe');
   WHEN PENDIENTE_MAYOR THEN
      DBMS_OUTPUT.PUT_LINE('Este puerto tiene una pendiente mayor que 7');

END;
```

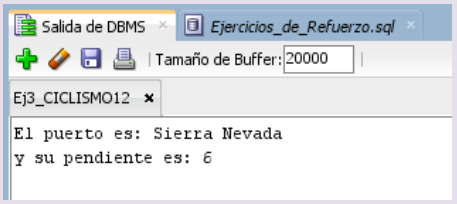



```
-- bloque anónimo

DECLARE
   V_NOMBRE_PUERTO PUERTO.NOMPUERTO%TYPE;
   V_PENDIENTE1 PUERTO.PENDIENTE%TYPE;

BEGIN
   V_NOMBRE_PUERTO := '&NOMPUERTO';

   V_PENDIENTE1 := f_devolverPendientePuerto(V_NOMBRE_PUERTO);

   DBMS_OUTPUT.PUT_LINE('El puerto es: ' || V_NOMBRE_PUERTO);
   DBMS_OUTPUT.PUT_LINE('y su pendiente es: ' || V_PENDIENTE1);

END;
```



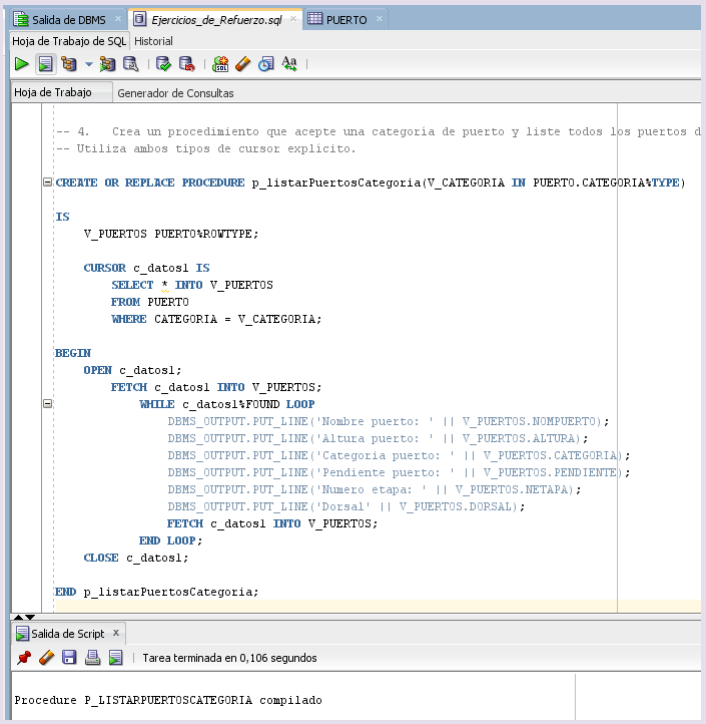


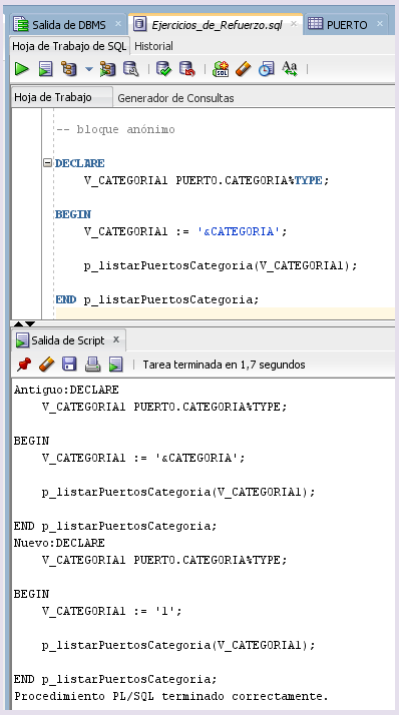
Ejemplo con "Navacerrada"

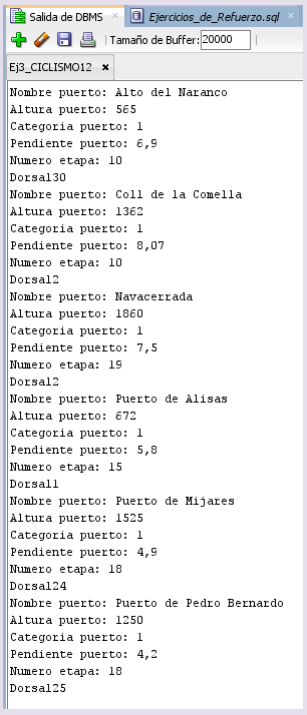


4. Crea un procedimiento que acepte una categoría de puerto y liste todos los puertos de esa categoría. Utiliza ambos tipos de cursor explícito.

```
CREATE OR REPLACE PROCEDURE p_listarPuertosCategoria(V_CATEGORIA IN PUERTO.CATEGORIA%TYPE)

IS
   V_PUERTOS PUERTO%ROWTYPE;

   CURSOR c_datos1 IS
      SELECT * INTO V_PUERTOS
      FROM PUERTO
      WHERE CATEGORIA = V_CATEGORIA;

BEGIN
   OPEN c_datos1;
      FETCH c_datos1 INTO V_PUERTOS;
         WHILE c_datos1%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE('Nombre puerto: ' ||
V_PUERTOS.NOMPUERTO);
            DBMS_OUTPUT.PUT_LINE('Altura puerto: ' ||
V_PUERTOS.ALTURA);
            DBMS_OUTPUT.PUT_LINE('Categoria puerto: ' ||
V_PUERTOS.CATEGORIA);
            DBMS_OUTPUT.PUT_LINE('Pendiente puerto: ' ||
V_PUERTOS.PENDIENTE);
            DBMS_OUTPUT.PUT_LINE('Numero etapa: ' ||
V_PUERTOS.NETAPA);
            DBMS_OUTPUT.PUT_LINE('Dorsal' ||
V_PUERTOS.DORSAL);
            FETCH c_datos1 INTO V_PUERTOS;
         END LOOP;
   CLOSE c_datos1;

END p_listarPuertosCategoria;

-- bloque anónimo

DECLARE
   V_CATEGORIA1 PUERTO.CATEGORIA%TYPE;

BEGIN
   V_CATEGORIA1 := '&CATEGORIA';

   p_listarPuertosCategoria(V_CATEGORIA1);

END p_listarPuertosCategoria;
```

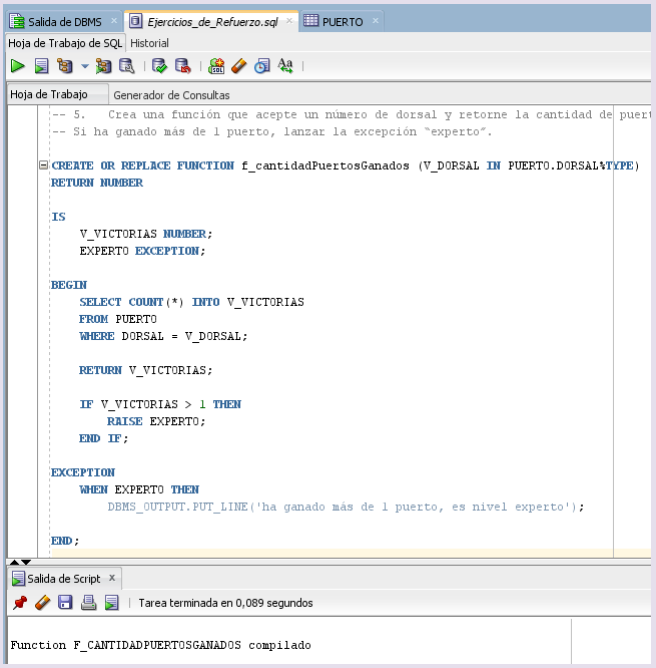




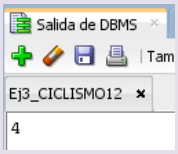


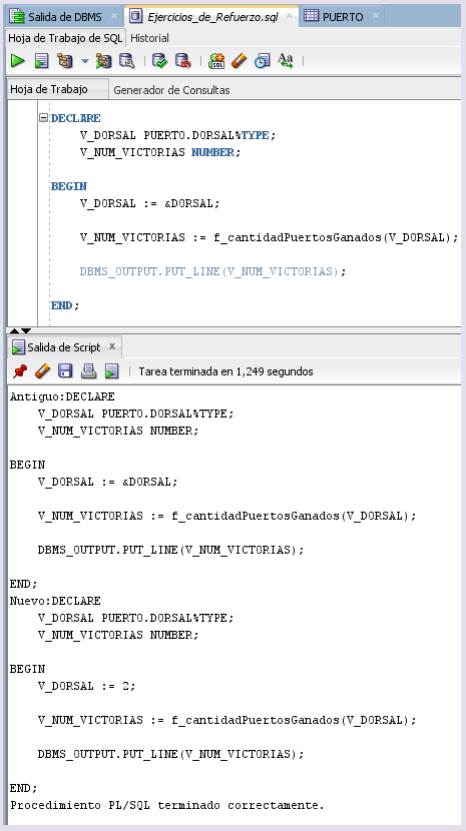
5. Crea una función que acepte un número de dorsal y retorne la cantidad de puertos que ha ganado. Si ha ganado más de 1 puerto, lanzar la excepción “experto”.

```
CREATE OR REPLACE FUNCTION f_cantidadPuertosGanados (V_DORSAL IN PUERTO.DORSAL%TYPE)
RETURN NUMBER

IS
   V_VICTORIAS NUMBER;
   EXPERTO EXCEPTION;

BEGIN
   SELECT COUNT(*) INTO V_VICTORIAS
   FROM PUERTO
   WHERE DORSAL = V_DORSAL;

   RETURN V_VICTORIAS;

   IF V_VICTORIAS > 1 THEN
      RAISE EXPERTO;
   END IF;

EXCEPTION
   WHEN EXPERTO THEN
      DBMS_OUTPUT.PUT_LINE('ha ganado más de 1 puerto, es nivel experto');

END;
```



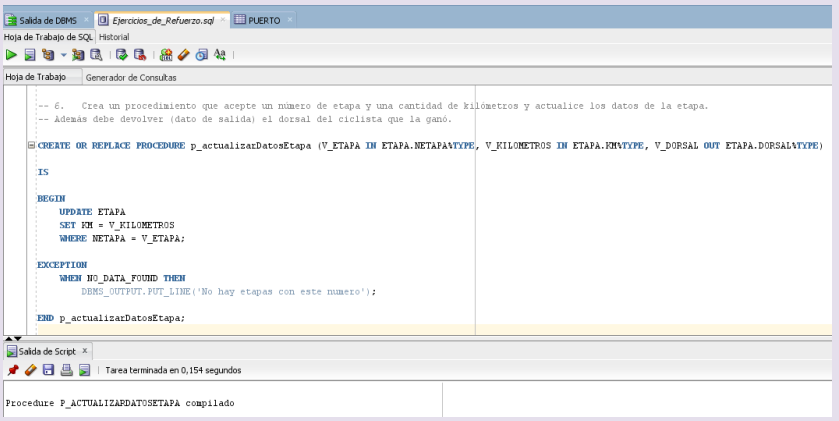

```
-- bloque anónimo

DECLARE
   V_DORSAL PUERTO.DORSAL%TYPE;
   V_NUM_VICTORIAS NUMBER;

BEGIN
   V_DORSAL := &DORSAL;

   V_NUM_VICTORIAS := f_cantidadPuertosGanados(V_DORSAL);

   DBMS_OUTPUT.PUT_LINE(V_NUM_VICTORIAS);

END;
```



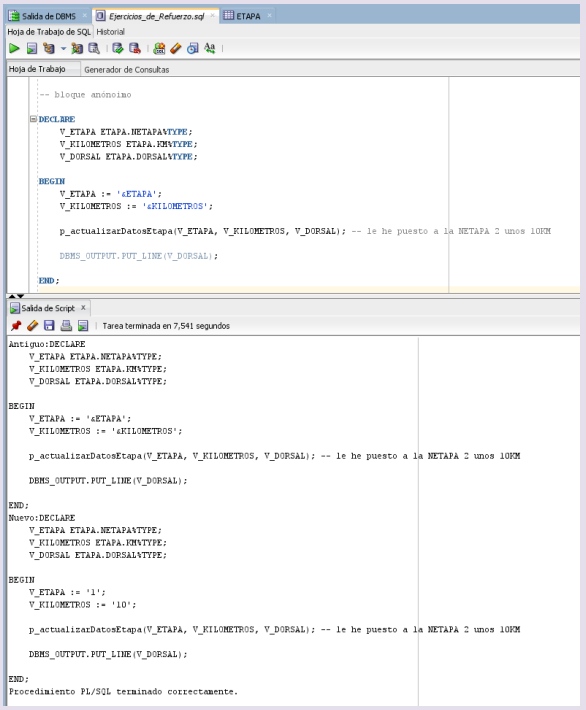


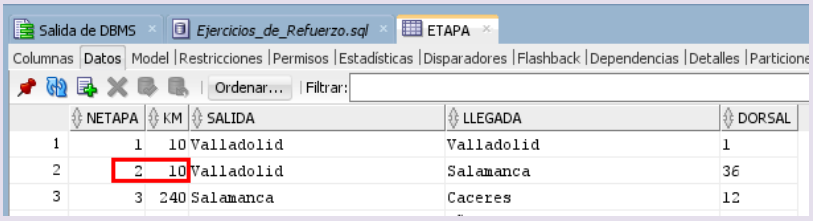
6. Crea un procedimiento que acepte un número de etapa y una cantidad de kilómetros y actualice los datos de la etapa. Además debe devolver (dato de salida) el dorsal del ciclista que la ganó.

```
CREATE OR REPLACE PROCEDURE p_actualizarDatosEtapa (V_ETAPA IN ETAPA.NETAPA%TYPE,
V_KILOMETROS IN ETAPA.KM%TYPE, V_DORSAL OUT ETAPA.DORSAL%TYPE)

IS

BEGIN
   UPDATE ETAPA
   SET KM = V_KILOMETROS
   WHERE NETAPA = V_ETAPA;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No hay etapas
con este numero');

END p_actualizarDatosEtapa;
```



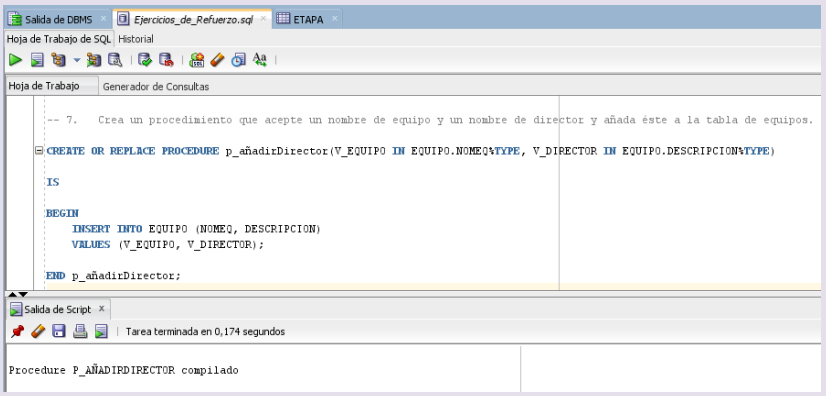

```
-- bloque anónoimo

DECLARE
   V_ETAPA ETAPA.NETAPA%TYPE;
   V_KILOMETROS ETAPA.KM%TYPE;
   V_DORSAL ETAPA.DORSAL%TYPE;

BEGIN
   V_ETAPA := '&ETAPA';
   V_KILOMETROS := '&KILOMETROS';

   p_actualizarDatosEtapa(V_ETAPA, V_KILOMETROS, V_DORSAL);

   DBMS_OUTPUT.PUT_LINE(V_DORSAL);

END;
```

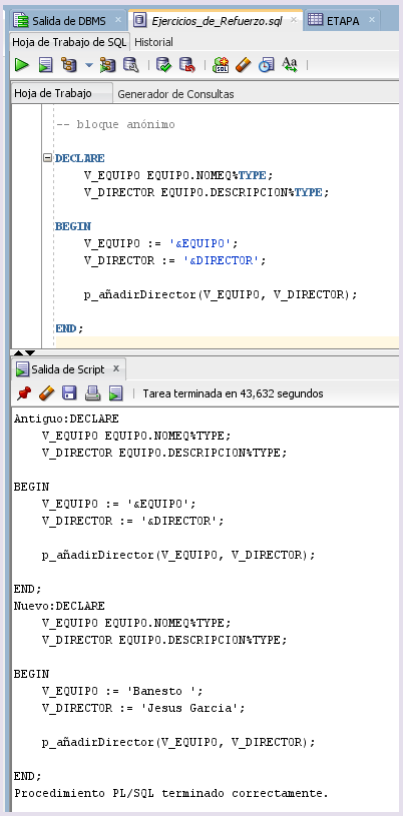




7. Crea un procedimiento que acepte un nombre de equipo y un nombre de director y añada éste a la tabla de equipos.

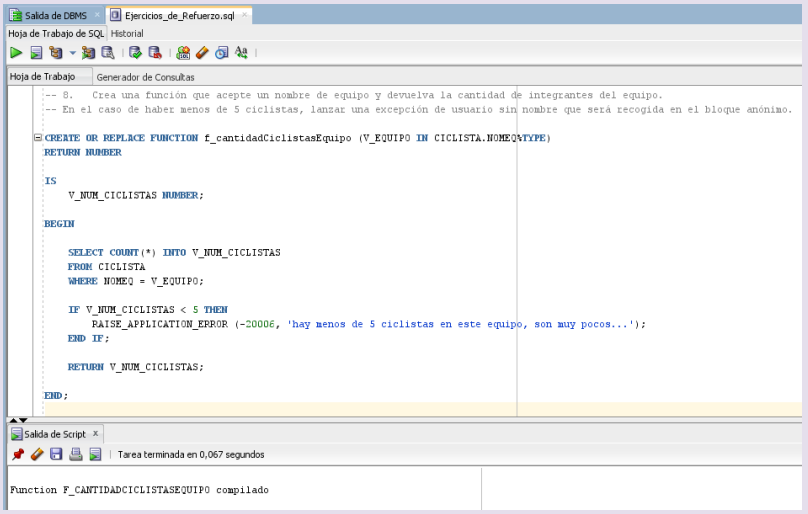
```
CREATE OR REPLACE PROCEDURE p_añadirDirector(V_EQUIPO IN EQUIPO.NOMEQ%TYPE, V_DIRECTOR IN
EQUIPO.DESCRIPCION%TYPE)

IS

BEGIN
   INSERT INTO EQUIPO (NOMEQ, DESCRIPCION)
   VALUES (V_EQUIPO, V_DIRECTOR);

END p_añadirDirector;

-- bloque anónimo

DECLARE
   V_EQUIPO EQUIPO.NOMEQ%TYPE;
   V_DIRECTOR EQUIPO.DESCRIPCION%TYPE;

BEGIN
   V_EQUIPO := '&EQUIPO';
   V_DIRECTOR := '&DIRECTOR';

   p_añadirDirector(V_EQUIPO, V_DIRECTOR);

END;
```



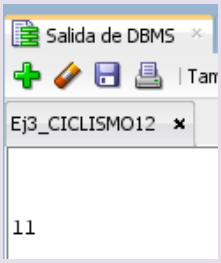


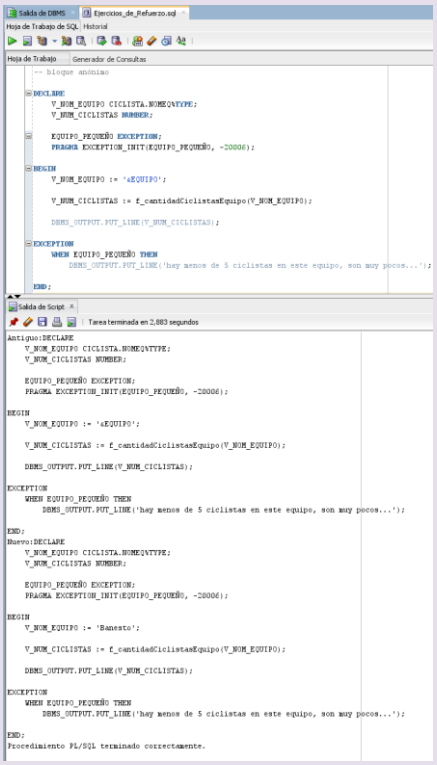
8. Crea una función que acepte un nombre de equipo y devuelva la cantidad de integrantes del equipo. En el caso de haber menos de 5 ciclistas, lanzar una excepción de usuario sin nombre que será recogida en el bloque anónimo.

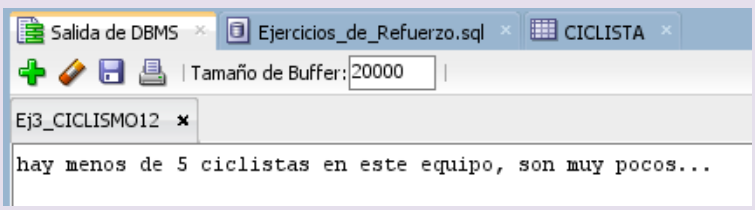
```
CREATE OR REPLACE FUNCTION f_cantidadCiclistasEquipo (V_EQUIPO IN CICLISTA.NOMEQ%TYPE)
RETURN NUMBER

IS
   V_NUM_CICLISTAS NUMBER;

BEGIN

   SELECT COUNT(*) INTO V_NUM_CICLISTAS
   FROM CICLISTA
   WHERE NOMEQ = V_EQUIPO;

   IF V_NUM_CICLISTAS < 5 THEN
      RAISE_APPLICATION_ERROR (-20006, 'hay
menos de 5 ciclistas en este equipo, son muy pocos...');
   END IF;

   RETURN V_NUM_CICLISTAS;

END;
```




```
-- bloque anónimo

DECLARE
   V_NOM_EQUIPO CICLISTA.NOMEQ%TYPE;
   V_NUM_CICLISTAS NUMBER;

   EQUIPO_PEQUEÑO EXCEPTION;
   PRAGMA EXCEPTION_INIT(EQUIPO_PEQUEÑO, -20006);

BEGIN
   V_NOM_EQUIPO := '&EQUIPO';

   V_NUM_CICLISTAS := f_cantidadCiclistasEquipo(V_NOM_EQUIPO);

   DBMS_OUTPUT.PUT_LINE(V_NUM_CICLISTAS);

EXCEPTION
   WHEN EQUIPO_PEQUEÑO THEN
      DBMS_OUTPUT.PUT_LINE('hay menos de 5 ciclistas en este equipo, son muy
pocos...');

END;
```



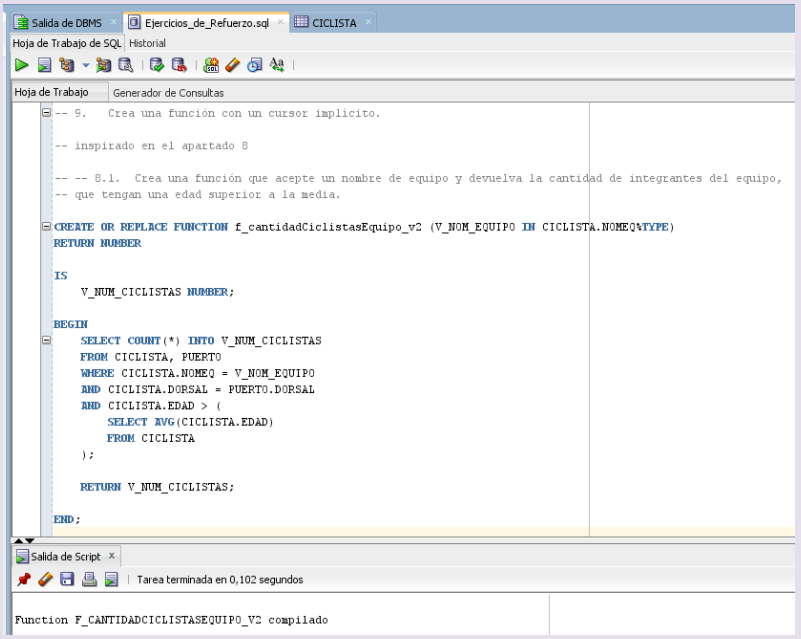


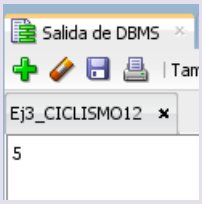


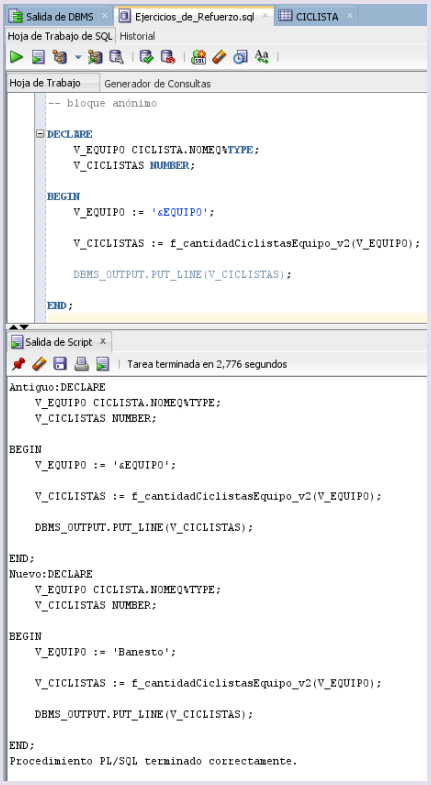
Ejemplo con "Amore Vita"

9. Crea una función con un cursor implícito.

```
-- inspirado en el apartado 8

-- -- 8.1. Crea una función que acepte un nombre de equipo y devuelva la cantidad de integrantes del equipo,
-- que tengan una edad superior a la media.

CREATE OR REPLACE FUNCTION f_cantidadCiclistasEquipo_v2 (V_NOM_EQUIPO IN
CICLISTA.NOMEQ%TYPE)
RETURN NUMBER

IS
   V_NUM_CICLISTAS NUMBER;

BEGIN
   SELECT COUNT(*) INTO V_NUM_CICLISTAS
   FROM CICLISTA, PUERTO
   WHERE CICLISTA.NOMEQ = V_NOM_EQUIPO
   AND CICLISTA.DORSAL = PUERTO.DORSAL
   AND CICLISTA.EDAD > (
      SELECT AVG(CICLISTA.EDAD)
      FROM CICLISTA
   );

   RETURN V_NUM_CICLISTAS;

END;

-- bloque anónimo

DECLARE
   V_EQUIPO CICLISTA.NOMEQ%TYPE;
   V_CICLISTAS NUMBER;

BEGIN
   V_EQUIPO := '&EQUIPO';

   V_CICLISTAS := f_cantidadCiclistasEquipo_v2(V_EQUIPO);

   DBMS_OUTPUT.PUT_LINE(V_CICLISTAS);

END;
```







10. Crea un procedimiento con un cursor explícito. Escríbelo con los dos tipos de cursor explícito posibles.

```
-- Replico el ejer 4 con bucle FOR

-- 4.	Crea un procedimiento que acepte una categoría de puerto y liste todos los puertos de esa categoría.
-- Utiliza ambos tipos de cursor explícito.

CREATE OR REPLACE PROCEDURE p_listarPuertosCategoria_v2(V_CATEGORIA IN
PUERTO.CATEGORIA%TYPE)

IS
   V_PUERTOS PUERTO%ROWTYPE;

   CURSOR c_datos1 IS
      SELECT * INTO V_PUERTOS
      FROM PUERTO
      WHERE CATEGORIA = V_CATEGORIA;

BEGIN
   FOR V_PUERTOS IN c_datos1 LOOP
      DBMS_OUTPUT.PUT_LINE('Nombre puerto: ' ||
V_PUERTOS.NOMPUERTO);
      DBMS_OUTPUT.PUT_LINE('Altura puerto: ' ||
V_PUERTOS.ALTURA);
      DBMS_OUTPUT.PUT_LINE('Categoria puerto: ' ||
V_PUERTOS.CATEGORIA);
      DBMS_OUTPUT.PUT_LINE('Pendiente puerto: ' ||
V_PUERTOS.PENDIENTE);
      DBMS_OUTPUT.PUT_LINE('Numero etapa: ' ||
V_PUERTOS.NETAPA);
      DBMS_OUTPUT.PUT_LINE('Dorsal' ||
V_PUERTOS.DORSAL);
   END LOOP;

END p_listarPuertosCategoria_v2;

-- bloque anónimo

DECLARE
   V_CATEGORIA1 PUERTO.CATEGORIA%TYPE;

BEGIN
   V_CATEGORIA1 := '&CATEGORIA';

   p_listarPuertosCategoria_v2(V_CATEGORIA1);

END p_listarPuertosCategoria_v2;
```