

TAREA TEMA 6 BAE

Autor: Derimán Tejera Fumero.

Fecha: 20/12/2023

Grupo: DAW Semi B.



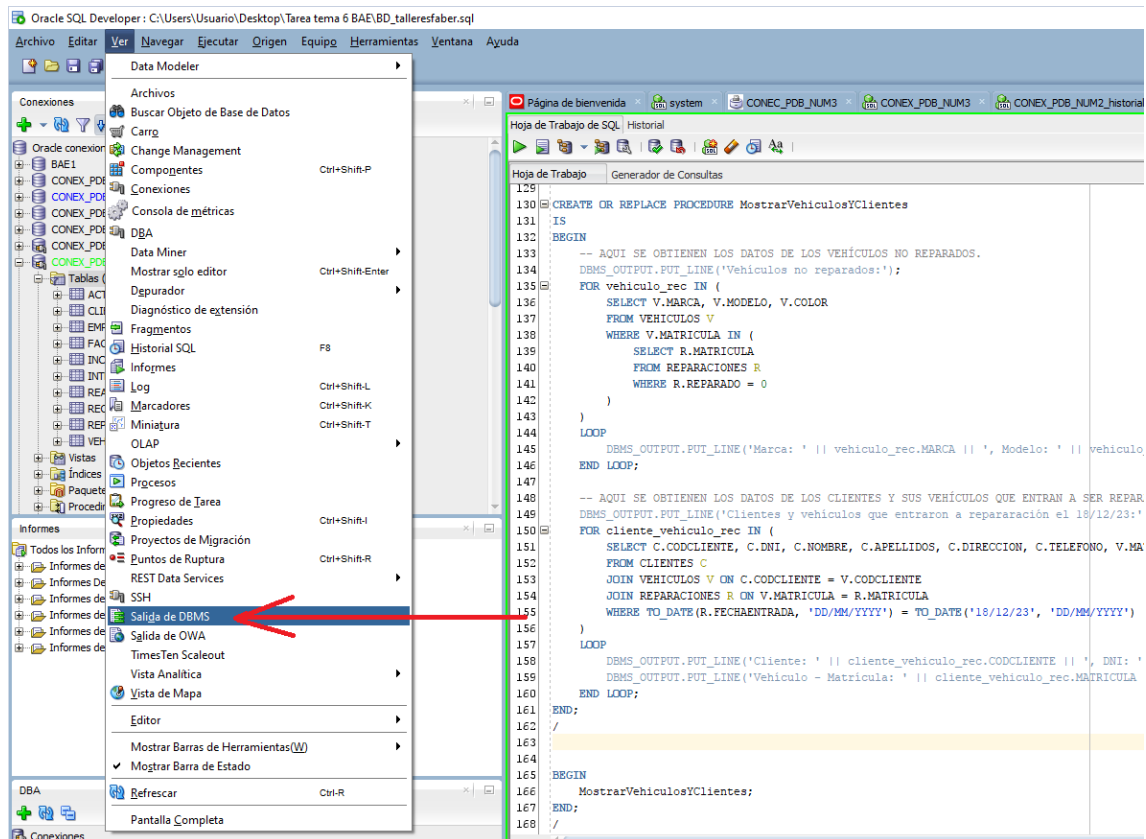
INDICE

Base de datos Talleres Faber	3
1.- Crear un procedimiento que muestre los vehículos (marca, modelo y color) que no estén reparados y los datos de los clientes y vehículos que han entrado a reparar hoy. (En nuestro caso ninguno).	3
2.- Crear una función que actualice el estado de las reparaciones que estén finalizadas en una fecha que se indique y que devuelva cuantas reparaciones han finalizado en esa fecha. 7	
3.-Un listado con dos columnas: en la primera, en mayúsculas apellidos y nombre de todos los clientes (entre los apellidos y el nombre incluir una coma como separador) y en la segunda, la ciudad en la que cada cliente tiene su domicilio (únicamente la ciudad, no la dirección).	9
4.- Creación de funciones:	11
Base de datos Historial Laboral	18
5.- Crear un procedimiento que realice un listado con los nombres y fechas, de todos los empleados y departamentos por los que ha pasado ordenado por fecha. Realizar la comprobación.	18
6.- Crear un procedimiento que actualice los estudios de un empleado (pasando como parámetros el dni del empleado, nombre de la universidad, año, grado y especialidad). En caso de error enviar un mensaje SIGNAL.	21
7.- Tabla de cambios datos empleados y trigger:	22
8.- Trigger y procedimiento:	25

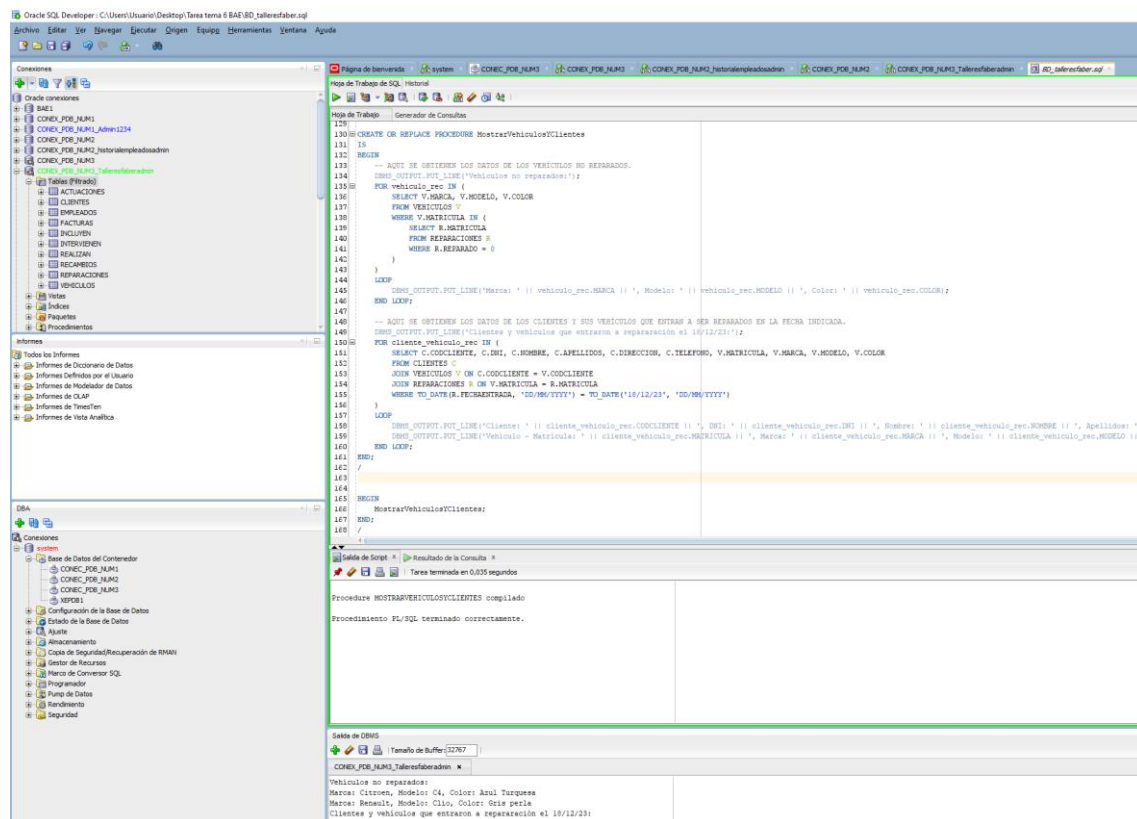
Base de datos Talleres Faber

1.- Crear un procedimiento que muestre los vehículos (marca, modelo y color) que no estén reparados y los datos de los clientes y vehículos que han entrado a reparar hoy. (En nuestro caso ninguno).

Antes de nada, hay que activar la opción “Salida de DBMS” para poder visualizar los resultados del procedimiento:



Captura del procedimiento funcionando:



Comandos:

```
CREATE OR REPLACE PROCEDURE MostrarVehiculosYClientes
IS
BEGIN
    -- AQUÍ SE OBTIENEN LOS DATOS DE LOS VEHÍCULOS NO REPARADOS.
    DBMS_OUTPUT.PUT_LINE('Vehículos no reparados:');
    FOR vehiculo_rec IN (
        SELECT V.MARCA, V.MODELO, V.COLOR
        FROM VEHICULOS V
        WHERE V.MATRICULA IN (
            SELECT R.MATRICULA
            FROM REPARACIONES R
            WHERE R.REPARADO = 0
        )
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Marca: ' || vehiculo_rec.MARCA || ',
Modelo: ' || vehiculo_rec.MODELO || ', Color: ' ||
vehiculo_rec.COLOR);
    END LOOP;

    -- AQUÍ SE OBTIENEN LOS DATOS DE LOS CLIENTES Y SUS VEHÍCULOS
QUE ENTRAN A SER REPARADOS EN LA FECHA INDICADA.
    DBMS_OUTPUT.PUT_LINE('Clientes y vehículos que entraron a
reparación el 18/12/23:');
    FOR cliente_vehiculo_rec IN (
```

```

        SELECT C.CODCLIENTE, C.DNI, C.NOMBRE, C.APELLIDOS,
        C.DIRECCION, C.TELEFONO, V.MATRICULA, V.MARCA, V.MODELO, V.COLOR
        FROM CLIENTES C
        JOIN VEHICULOS V ON C.CODCLIENTE = V.CODCLIENTE
        JOIN REPARACIONES R ON V.MATRICULA = R.MATRICULA
        WHERE TO_DATE(R.FECHAENTRADA, 'DD/MM/YYYY') =
        TO_DATE('18/12/23', 'DD/MM/YYYY')
    )
    LOOP
        DBMS_OUTPUT.PUT_LINE('Cliente: ' ||
        cliente_vehiculo_rec.CODCLIENTE || ', DNI: ' ||
        cliente_vehiculo_rec.DNI || ', Nombre: ' ||
        cliente_vehiculo_rec.NOMBRE || ', Apellidos: ' ||
        cliente_vehiculo_rec.APELLIDOS || ', Dirección: ' ||
        cliente_vehiculo_rec.DIRECCION || ', Teléfono: ' ||
        cliente_vehiculo_rec.TELEFONO);
        DBMS_OUTPUT.PUT_LINE('Vehículo - Matrícula: ' ||
        cliente_vehiculo_rec.MATRICULA || ', Marca: ' ||
        cliente_vehiculo_rec.MARCA || ', Modelo: ' ||
        cliente_vehiculo_rec.MODELO || ', Color: ' ||
        cliente_vehiculo_rec.COLOR);
    END LOOP;
END;
/

```

Y luego, para poder mostrar los resultados, ejecutamos:

Comandos:

```

BEGIN
    MostrarVehiculosYClientes;
END;
/

```

Salida de Script x

Resultado de la Consulta x

Tarea terminada en 0,035 segundos

Procedure MOSTRARVEHICULOSYCLIENTES compilado
Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

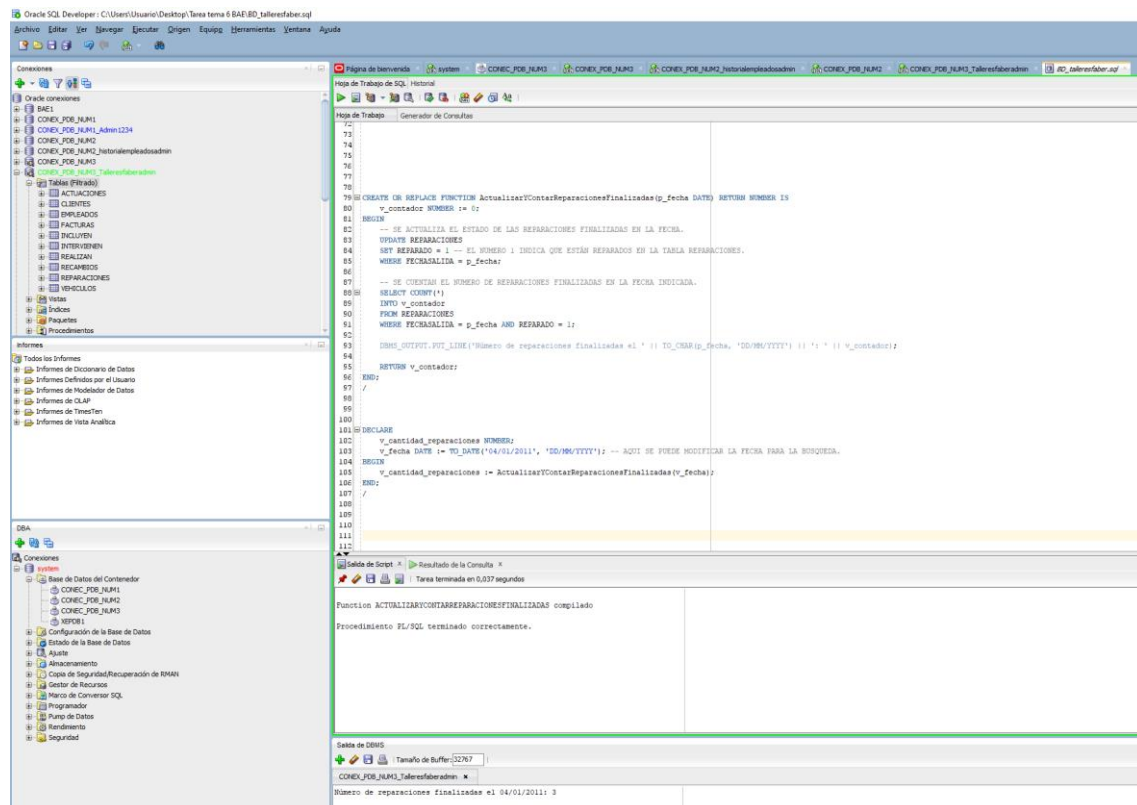
Tamaño de Buffer: 32767

CONEX_PDB_NUM3_Talleresfaberadmin x

Vehículos no reparados:
Marca: Citroen, Modelo: C4, Color: Azul Turquesa
Marca: Renault, Modelo: Clio, Color: Gris perla
Clientes y vehículos que entraron a reparación el 18/12/23:

No se muestra ningún vehículo en esa fecha porque no hay ninguno en la base de datos, pero e ha probado con una fecha de un vehículo que si existe en la base de datos y funciona.

2.- Crear una función que actualice el estado de las reparaciones que estén finalizadas en una fecha que se indique y que devuelva cuantas reparaciones han finalizado en esa fecha.



Comandos:

```
CREATE OR REPLACE FUNCTION
ActualizarYContarReparacionesFinalizadas(p_fecha DATE) RETURN NUMBER
IS
    v_contador NUMBER := 0;
BEGIN
    -- SE ACTUALIZA EL ESTADO DE LAS REPARACIONES FINALIZADAS EN LA
    FECHA.
    UPDATE REPARACIONES
    SET REPARADO = 1 -- EL NUMERO 1 INDICA QUE ESTÁN REPARADOS EN LA
    TABLA REPARACIONES.
    WHERE FECHASALIDA = p_fecha;

    -- SE CUENTAN EL NUMERO DE REPARACIONES FINALIZADAS EN LA FECHA
    INDICADA.
    SELECT COUNT(*)
    INTO v_contador
    FROM REPARACIONES
    WHERE FECHASALIDA = p_fecha AND REPARADO = 1;

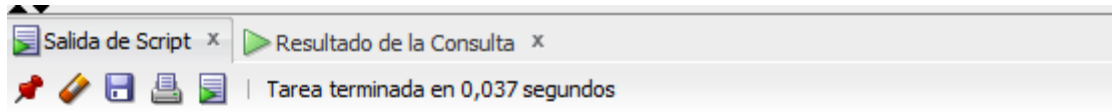
    DBMS_OUTPUT.PUT_LINE('Número de reparaciones finalizadas el ' ||
    TO_CHAR(p_fecha, 'DD/MM/YYYY') || ': ' || v_contador);
```

```
        RETURN v_contador;  
END;  
/
```

Y luego, para poder mostrar los resultados, ejecutamos:

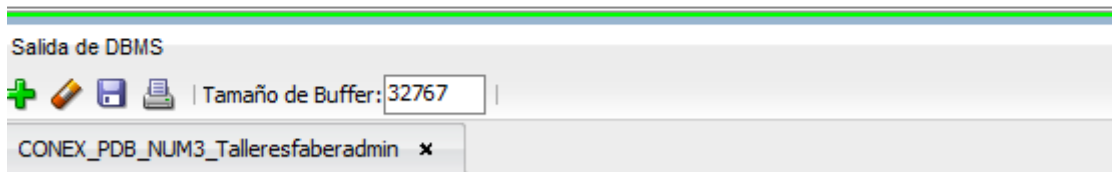
Comandos:

```
DECLARE  
    v_cantidad_reparaciones NUMBER;  
    v_fecha DATE := TO_DATE('04/01/2011', 'DD/MM/YYYY'); -- AQUI SE  
    PUEDE MODIFICAR LA FECHA PARA LA BUSQUEDA.  
BEGIN  
    v_cantidad_reparaciones :=  
    ActualizarYContarReparacionesFinalizadas(v_fecha);  
END;  
/
```



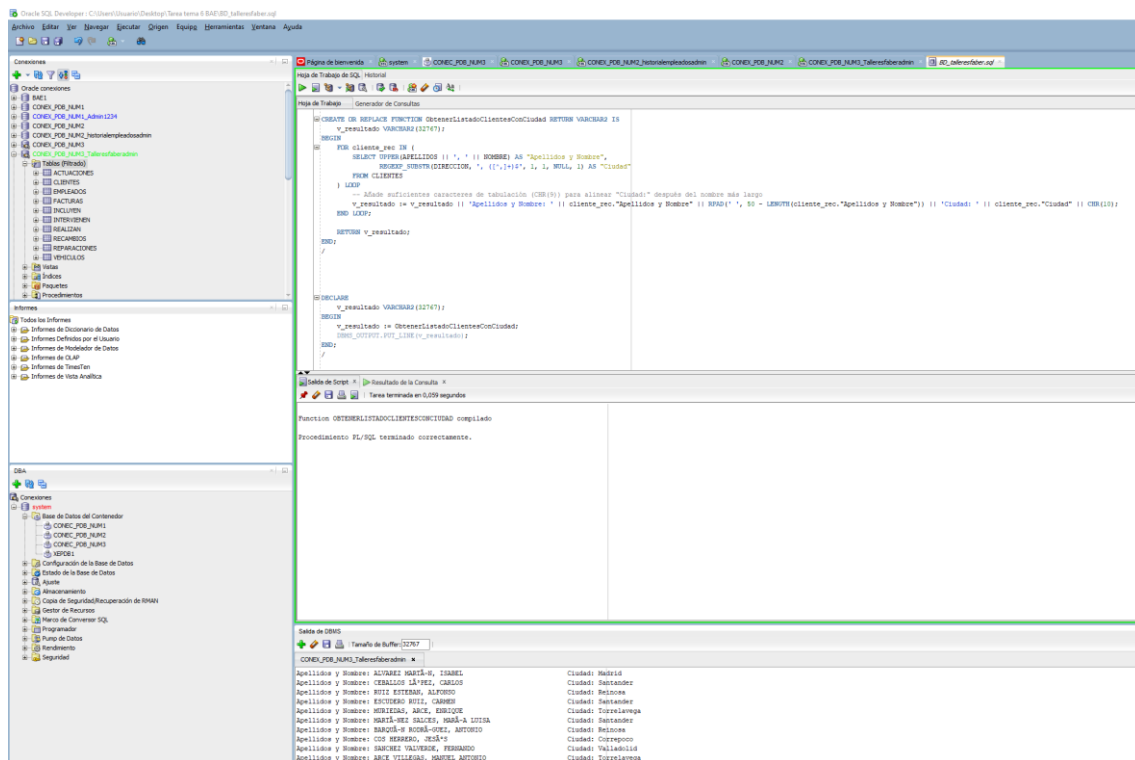
Function ACTUALIZARYCONTARREPARACIONESFINALIZADAS compilado

Procedimiento PL/SQL terminado correctamente.



Número de reparaciones finalizadas el 04/01/2011: 3

3.-Un listado con dos columnas: en la primera, en mayúsculas apellidos y nombre de todos los clientes (entre los apellidos y el nombre incluir una coma como separador) y en la segunda, la ciudad en la que cada cliente tiene su domicilio (únicamente la ciudad, no la dirección).



Comandos:

```
CREATE OR REPLACE FUNCTION ObtenerListadoClientesConCiudad RETURN
VARCHAR2 IS
    v_resultado VARCHAR2(32767);
BEGIN
    FOR cliente_rec IN (
        SELECT UPPER(APELLIDOS || ', ' || NOMBRE) AS "Apellidos y
        Nombre",
                REGEXP_SUBSTR(DIRECCION, ', ([^,]+)$', 1, 1, NULL, 1)
        AS "Ciudad"
        FROM CLIENTES
    ) LOOP
        -- Añade suficientes caracteres de tabulación (CHR(9)) para
        alinear "Ciudad:" después del nombre más largo
        v_resultado := v_resultado || 'Apellidos y Nombre: ' ||
        cliente_rec."Apellidos y Nombre" || RPAD(' ', 50 -
        LENGTH(cliente_rec."Apellidos y Nombre")) || 'Ciudad: ' ||
        cliente_rec."Ciudad" || CHR(10);
    END LOOP;

    RETURN v_resultado;
```

```
END;  
/
```

Y luego, para poder mostrar los resultados, ejecutamos:

Comandos:

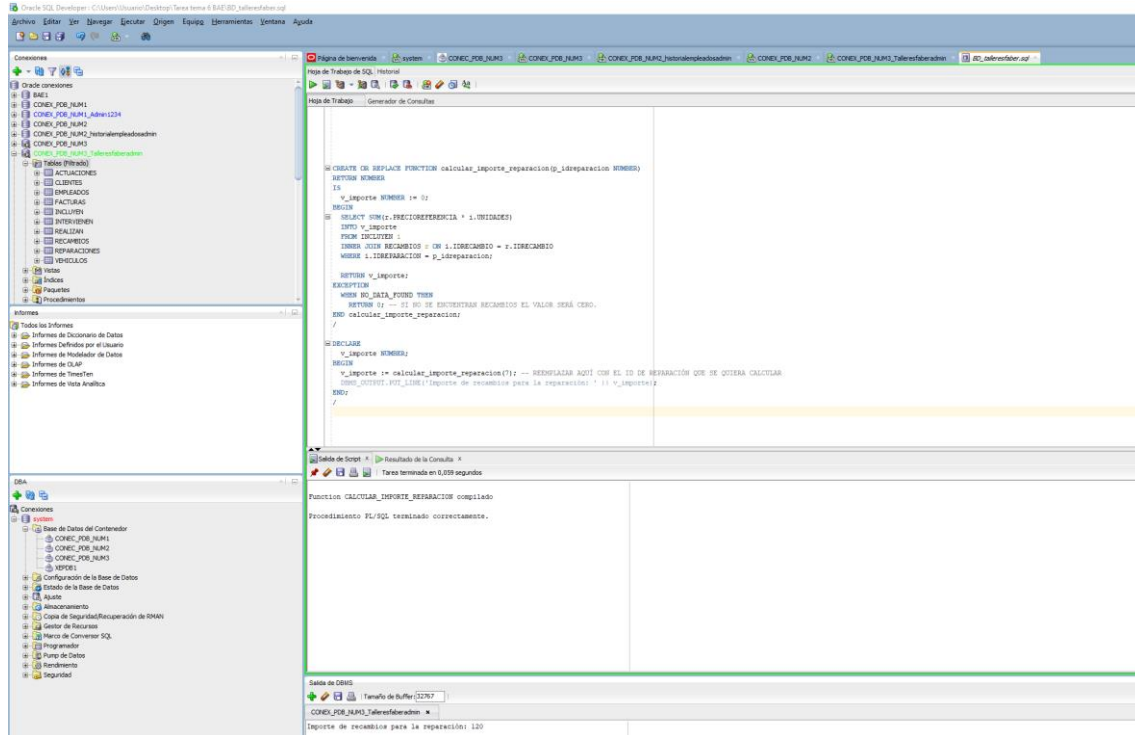
```
DECLARE  
    v_resultado VARCHAR2(32767);  
BEGIN  
    v_resultado := ObtenerListadoClientesConCiudad;  
    DBMS_OUTPUT.PUT_LINE(v_resultado);  
END;  
/
```

The screenshot shows the SQL Developer interface. The top pane, titled 'Resultado de la Consulta', displays the execution status: 'Function OBTENERLISTADOCLIENTESCONCIUDAD compilado' and 'Procedimiento PL/SQL terminado correctamente.' Below this, the 'Salida de DBMS' pane shows a table of results. The table has two columns: 'Apellidos y Nombre' and 'Ciudad'. The data is as follows:

Apellidos y Nombre	Ciudad
ALVAREZ MARTÃ-N, ISABEL	Madrid
CEBALLOS LÃ-PEZ, CARLOS	Santander
RUIZ ESTEBAN, ALFONSO	Reinosa
ESCUDERO RUIZ, CARMEN	Santander
MURIEDAS, ARCE, ENRIQUE	Torrelavega
MARTÃ-NEZ SALCES, MARÃ-A LUISA	Santander
BARQUÃ-N RODRÃ-GUEZ, ANTONIO	Reinosa
COS HERRERO, JESÃ-S	Correpoco
SANCHEZ VALVERDE, FERNANDO	Valladolid
ARCE VILLEGAS, MANUEL ANTONIO	Torrelavega

4.- Creación de funciones:

a) Diseña una función que calcule el importe de los recambios sustituidos en una reparación.



Comandos:

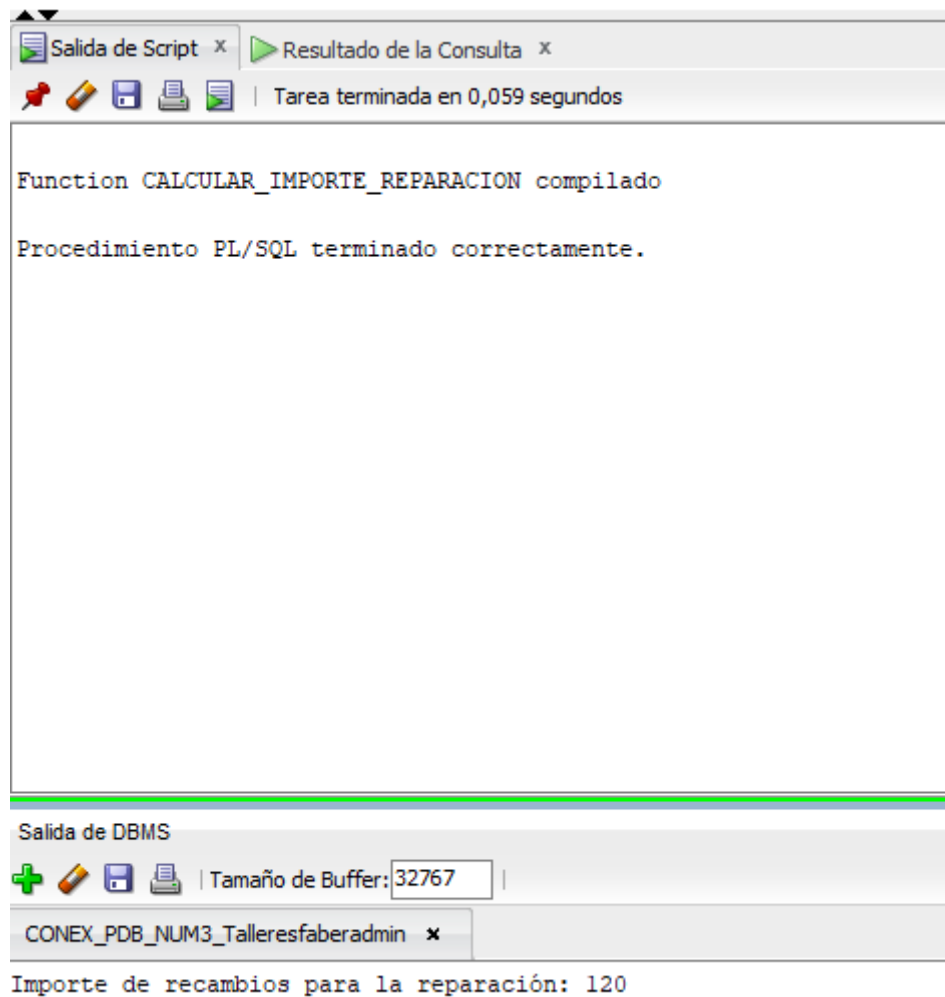
```
CREATE OR REPLACE FUNCTION
calcular_importe_reparacion(p_idreparacion NUMBER)
RETURN NUMBER
IS
    v_importe NUMBER := 0;
BEGIN
    SELECT SUM(r.PRECIOREFERENCIA * i.UNIDADES)
    INTO v_importe
    FROM INCLUYEN i
    INNER JOIN RECAMBIOS r ON i.IDRECAMBIO = r.IDRECAMBIO
    WHERE i.IDREPARACION = p_idreparacion;

    RETURN v_importe;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0; -- SI NO SE ENCUENTRAN RECAMBIOS EL VALOR SERÁ CERO.
END calcular_importe_reparacion;
/
```

Y luego, para poder mostrar los resultados, ejecutamos:

Comandos:

```
DECLARE
  v_importe NUMBER;
BEGIN
  v_importe := calcular_importe_reparacion(7); -- REEMPLAZAR AQUÍ
  CON EL ID DE REPARACIÓN QUE SE QUIERA CALCULAR
  DBMS_OUTPUT.PUT_LINE('Importe de recambios para la reparación: '
  || v_importe);
END;
/
```



The screenshot shows a SQL Developer window with two tabs: "Salida de Script" and "Resultado de la Consulta". The "Resultado de la Consulta" tab is active, displaying the output of the script execution. The output text is:

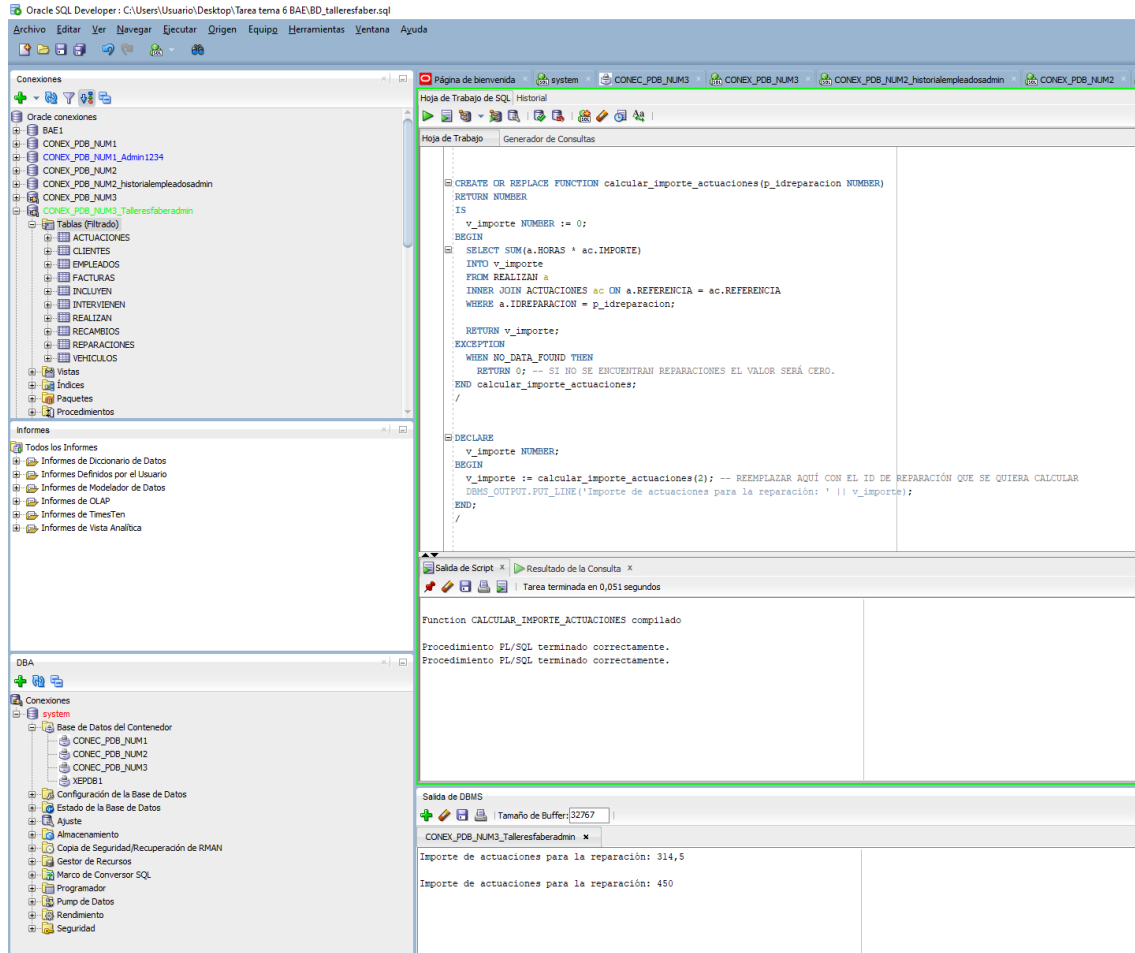
```
Function CALCULAR_IMPORTE_REPARACION compilado
Procedimiento PL/SQL terminado correctamente.
```

Below the main window, there is a "Salida de DBMS" section. It includes a toolbar with icons for adding, editing, saving, and printing, along with a "Tamaño de Buffer" field set to 32767. Below this, there is a tab labeled "CONEX_PDB_NUM3_Talleresfaberadmin". The output of the DBMS output is displayed below the tab:

```
Importe de recambios para la reparación: 120
```

b) Crear una función que devuelva el importe de las actuaciones que se llevan a cabo en una reparación (para calcular el importe multiplica las horas por el importe de cada actuación).

En ambas funciones Pasar como variable el Id de la reparación.



Comandos:

```
CREATE OR REPLACE FUNCTION
calcular_importe_actuaciones(p_idreparacion NUMBER)
RETURN NUMBER
IS
    v_importe NUMBER := 0;
BEGIN
    SELECT SUM(a.HORAS * ac.IMPORTE)
    INTO v_importe
    FROM REALIZAN a
    INNER JOIN ACTUACIONES ac ON a.REFERENCIA = ac.REFERENCIA
    WHERE a.IDREPARACION = p_idreparacion;

    RETURN v_importe;
EXCEPTION
```

```

    WHEN NO_DATA_FOUND THEN
        RETURN 0; -- SI NO SE ENCUENTRAN REPARACIONES EL VALOR SERÁ
CERO.
END calcular_importe_actuaciones;
/

```

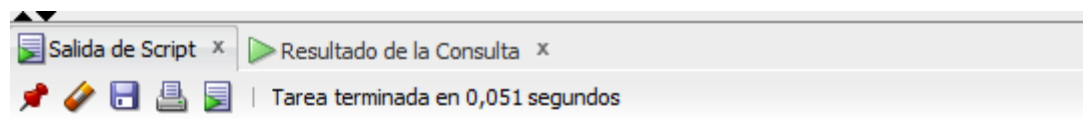
Y luego, para poder mostrar los resultados, ejecutamos:

Comandos:

```

DECLARE
    v_importe NUMBER;
BEGIN
    v_importe := calcular_importe_actuaciones(2); -- REEMPLAZAR AQUÍ
CON EL ID DE REPARACIÓN QUE SE QUIERA CALCULAR
    DBMS_OUTPUT.PUT_LINE('Importe de actuaciones para la reparación: '
|| v_importe);
END;
/

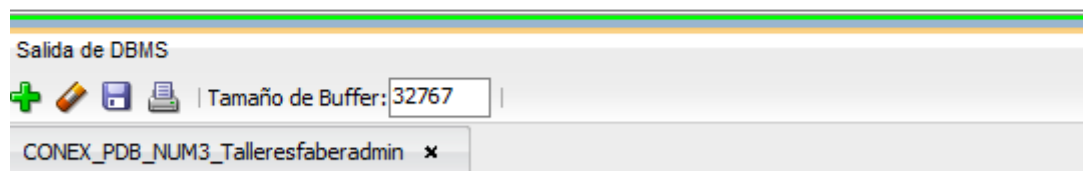
```



Function CALCULAR_IMPORTE_ACTUACIONES compilado

Procedimiento PL/SQL terminado correctamente.

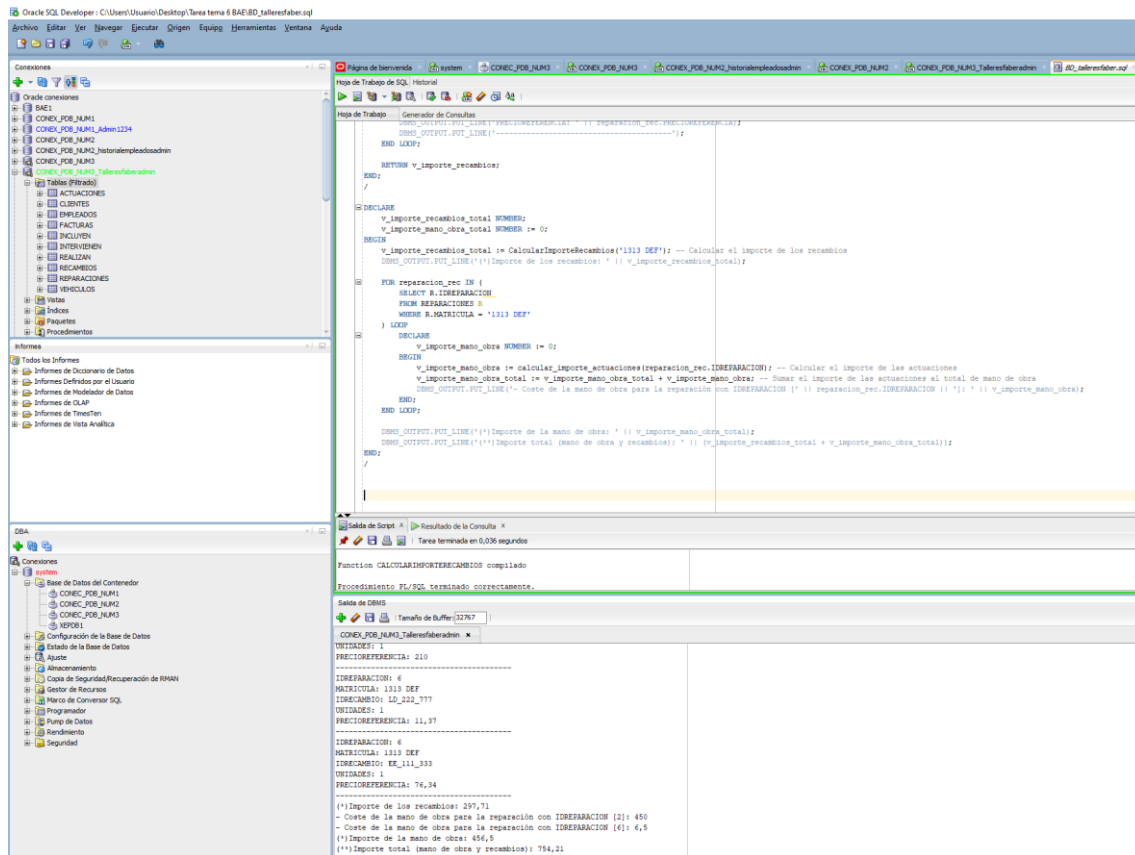
Procedimiento PL/SQL terminado correctamente.



Importe de actuaciones para la reparación: 314,5

Importe de actuaciones para la reparación: 450

c) Hacer una consulta que calcule el importe total (mano de obra y recambios) de las reparaciones que se le hayan realizado al vehículo de matrícula '1313 DEF'.



Comandos:

```

CREATE OR REPLACE FUNCTION
CalcularImporteRecambios(matricula_vehiculo VARCHAR2) RETURN NUMBER
IS
    v_importe_recambios NUMBER := 0;
BEGIN
    -- BUSCA LAS REPARACIONES ASOCIADAS A LA MATRICULA.
    FOR reparacion_rec IN (
        SELECT DISTINCT R.IDREPARACION, R.MATRICULA,
            RC.IDRECAMBIO, IC.UNIDADES, RC.PRECIOREFERENCIA
        FROM REPARACIONES R
        JOIN INCLUYEN IC ON R.IDREPARACION = IC.IDREPARACION
        JOIN RECAMBIOS RC ON IC.IDRECAMBIO = RC.IDRECAMBIO
        WHERE R.MATRICULA = matricula_vehiculo
    ) LOOP
        -- CALCULA EL IMPORTE DE CADA RECAMBIO Y LO SUMA AL IMPORTE
        TOTAL DE RECAMBIOS.
        v_importe_recambios := v_importe_recambios +
            (reparacion_rec.PRECIOREFERENCIA * reparacion_rec.UNIDADES);
    END LOOP;
END;

```

```

        -- MUESTRA LOS DATOS DE LA REPARACIÓN Y EL RECAMBIO.
        DBMS_OUTPUT.PUT_LINE('IDREPARACION: ' ||
reparacion_rec.IDREPARACION);
        DBMS_OUTPUT.PUT_LINE('MATRICULA: ' ||
reparacion_rec.MATRICULA);
        DBMS_OUTPUT.PUT_LINE('IDRECAMBIO: ' ||
reparacion_rec.IDRECAMBIO);
        DBMS_OUTPUT.PUT_LINE('UNIDADES: ' ||
reparacion_rec.UNIDADES);
        DBMS_OUTPUT.PUT_LINE('PRECIOREFERENCIA: ' ||
reparacion_rec.PRECIOREFERENCIA);
        DBMS_OUTPUT.PUT_LINE('-----
--');
    END LOOP;

    RETURN v_importe_recambios;
END;
/

DECLARE
    v_importe_recambios_total NUMBER;
    v_importe_manoobra_total NUMBER := 0;
BEGIN
    v_importe_recambios_total := CalcularImporteRecambios('1313
DEF'); -- Calcular el importe de los recambios
    DBMS_OUTPUT.PUT_LINE('(*)Importe de los recambios: ' ||
v_importe_recambios_total);

    FOR reparacion_rec IN (
        SELECT R.IDREPARACION
        FROM REPARACIONES R
        WHERE R.MATRICULA = '1313 DEF'
    ) LOOP
        DECLARE
            v_importe_manoobra NUMBER := 0;
        BEGIN
            v_importe_manoobra :=
calcular_importe_actuaciones(reparacion_rec.IDREPARACION); --
Calcular el importe de las actuaciones
            v_importe_manoobra_total := v_importe_manoobra_total +
v_importe_manoobra; -- Sumar el importe de las actuaciones al total
de mano de obra
            DBMS_OUTPUT.PUT_LINE('- Coste de la mano de obra para la
reparación con IDREPARACION [' || reparacion_rec.IDREPARACION || ']:
' || v_importe_manoobra);
        END;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('(*)Importe de la mano de obra: ' ||
v_importe_manoobra_total);
    DBMS_OUTPUT.PUT_LINE('(**)Importe total (mano de obra y
recambios): ' || (v_importe_recambios_total +
v_importe_manoobra_total));
END;

```




Salida de Script x

Resultado de la Consulta x

Tarea terminada en 0,036 segundos

Procedimiento PL/SQL terminado correctamente.

Function CALCULARIMPORTERECAMBIOS compilado

Procedimiento PL/SQL terminado correctamente.

Function CALCULARIMPORTERECAMBIOS compilado

Procedimiento PL/SQL terminado correctamente.

Function CALCULARIMPORTERECAMBIOS compilado

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

Tamaño de Buffer: 32767

CONEX_PDB_NUM3_Talleresfaberadmin x

PRECIOREFERENCIA: 210

IDREPARACION: 6

MATRICULA: 1313 DEF

IDRECAMBIO: LD_222_777

UNIDADES: 1

PRECIOREFERENCIA: 11,37

IDREPARACION: 6

MATRICULA: 1313 DEF

IDRECAMBIO: EE_111_333

UNIDADES: 1

PRECIOREFERENCIA: 76,34

(*)Importe de los recambios: 297,71

- Coste de la mano de obra para la reparación con IDREPARACION [2]: 450

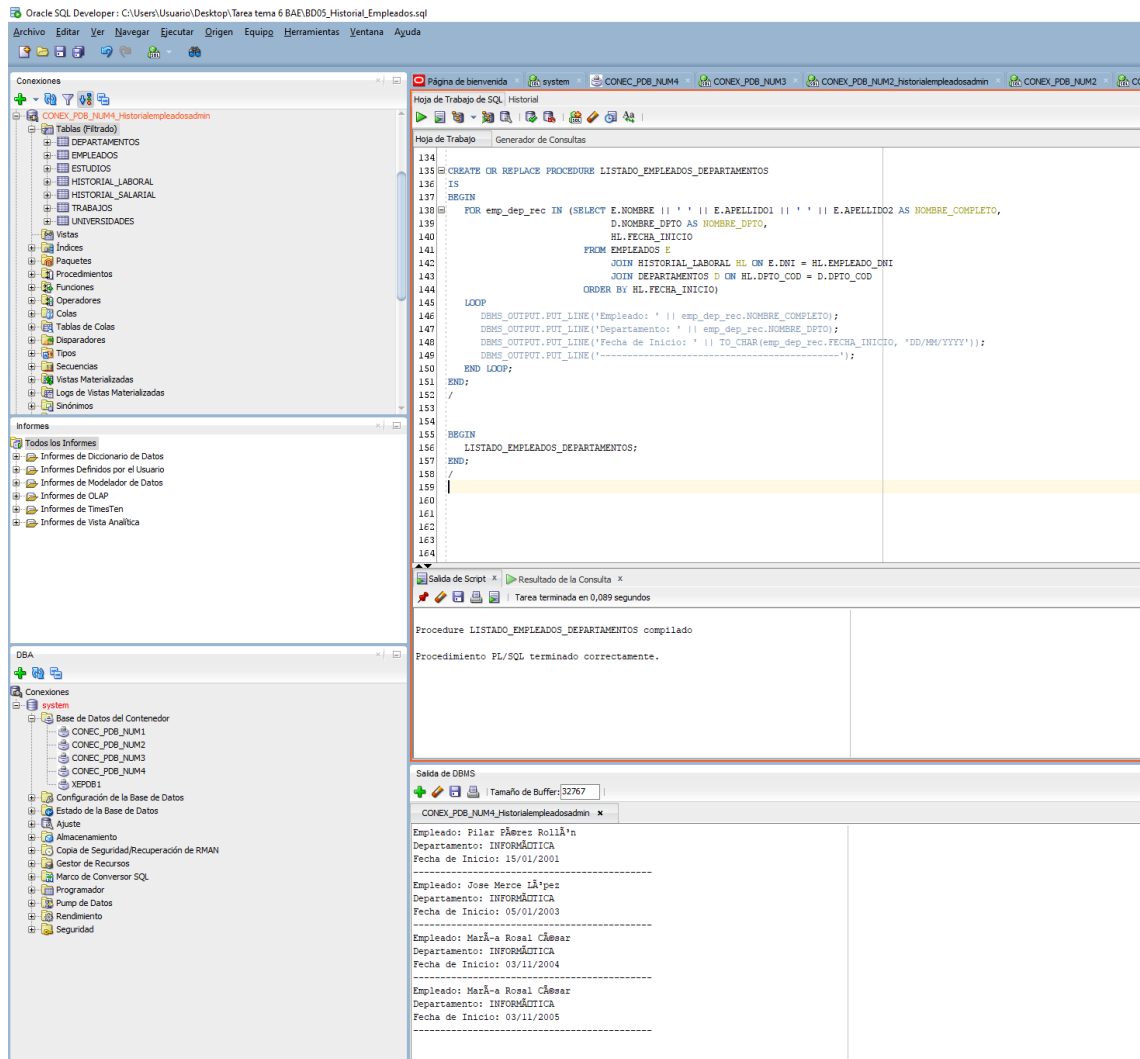
- Coste de la mano de obra para la reparación con IDREPARACION [6]: 6,5

(*)Importe de la mano de obra: 456,5

(**)Importe total (mano de obra y recambios): 754,21

Base de datos Historial Laboral

5.- Crear un procedimiento que realice un listado con los nombres y fechas, de todos los empleados y departamentos por los que ha pasado ordenado por fecha. Realizar la comprobación.



Comandos:

```
CREATE OR REPLACE PROCEDURE LISTADO_EMPLEADOS_DEPARTAMENTOS
IS
BEGIN
  FOR emp_dep_rec IN (SELECT E.NOMBRE || ' ' || E.APELLIDO1 || ' ' ||
                        || E.APELLIDO2 AS NOMBRE_COMPLETO,
                        D.NOMBRE_DPTO AS NOMBRE_DPTO,
                        HL.FECHA_INICIO
                        FROM EMPLEADOS E
```

```

                                JOIN HISTORIAL_LABORAL HL ON E.DNI =
HL.EMPLEADO_DNI
                                JOIN DEPARTAMENTOS D ON HL.DPTO_COD =
D.DPTO_COD
                                ORDER BY HL.FECHA_INICIO)
    LOOP
        DBMS_OUTPUT.PUT_LINE('Empleado: ' ||
emp_dep_rec.NOMBRE_COMPLETO);
        DBMS_OUTPUT.PUT_LINE('Departamento: ' ||
emp_dep_rec.NOMBRE_DPTO);
        DBMS_OUTPUT.PUT_LINE('Fecha de Inicio: ' ||
TO_CHAR(emp_dep_rec.FECHA_INICIO, 'DD/MM/YYYY'));
        DBMS_OUTPUT.PUT_LINE('-----
----');
    END LOOP;
END;
/

```

Y luego, para poder mostrar los resultados, ejecutamos:






Comandos:

```

BEGIN
    LISTADO_EMPLEADOS_DEPARTAMENTOS;
END;
/

```





Salida de Script x Resultado de la Consulta x

 | Tarea terminada en 0,089 segundos

Procedure LISTADO_EMPLEADOS_DEPARTAMENTOS compilado

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

 | Tamaño de Buffer: 32767 |

CONEX_PDB_NUM4_Historiaempleadosadmin x

Empleado: Pilar PÃ©rez RollÃ¡n
Departamento: INFORMÃ¡TICA
Fecha de Inicio: 15/01/2001

Empleado: Jose Merce LÃ¡pez
Departamento: INFORMÃ¡TICA
Fecha de Inicio: 05/01/2003

Empleado: MarÃ­a Rosal CÃ©sar
Departamento: INFORMÃ¡TICA
Fecha de Inicio: 03/11/2004

Empleado: MarÃ­a Rosal CÃ©sar
Departamento: INFORMÃ¡TICA
Fecha de Inicio: 03/11/2005

6.- Crear un procedimiento que actualice los estudios de un empleado (pasando como parámetros el dni del empleado, nombre de la universidad, año, grado y especialidad). En caso de error enviar un mensaje SIGNAL.

The screenshot shows the Oracle SQL Developer interface. The main window displays a PL/SQL procedure named `ACTUALIZAR_ESTUDIOS_CON_COMMIT`. The procedure takes five parameters: `p_DNI`, `p_NOMBRE_UNIV`, `p_AGNO`, `p_GRADO`, and `p_ESPECIALIDAD`. It uses a loop to update the `ESTUDIOS` table for each employee with the same DNI. The results pane shows the output of the procedure, displaying the updated data for three employees.

```

199
200 DECLARE
201   v_DNI NUMBER := 12345;
202   v_NOMBRE_UNIV VARCHAR2(25) := 'UNED';
203   v_AGNO NUMBER := 1992;
204   v_GRADO VARCHAR2(5) := 'MED';
205   v_ESPECIALIDAD VARCHAR2(20) := 'PRUEBAYUJYU';
206 BEGIN
207   -- Llamamos al procedimiento para actualizar los estudios
208   ACTUALIZAR_ESTUDIOS_CON_COMMIT(
209     p_DNI => v_DNI,
210     p_NOMBRE_UNIV => v_NOMBRE_UNIV,
211     p_AGNO => v_AGNO,
212     p_GRADO => v_GRADO,
213     p_ESPECIALIDAD => v_ESPECIALIDAD
214   );
215
216   -- Mostramos el contenido actualizado de la tabla ESTUDIOS
217   FOR estudios_rec IN (SELECT *
218                        FROM ESTUDIOS
219                        WHERE EMPLEADO_DNI = v_DNI)
220   LOOP
221     DBMS_OUTPUT.PUT_LINE('DNI: ' || estudios_rec.EMPLEADO_DNI);
222     DBMS_OUTPUT.PUT_LINE('Universidad: ' || estudios_rec.UNIVERSIDAD);
223     DBMS_OUTPUT.PUT_LINE('Año: ' || estudios_rec.AGNO);
224     DBMS_OUTPUT.PUT_LINE('Grado: ' || estudios_rec.GRADO);
225     DBMS_OUTPUT.PUT_LINE('Especialidad: ' || estudios_rec.ESPECIALIDAD);
226   END LOOP;
227 END;
228 /
229

```

EMPLEADO_DNI	UNIVERSIDAD	AGNO	GRADO	ESPECIALIDAD
1 12345	1 1992	MED	PRUEBAYUJYU	
2 22222	1 1998	SUP	ING INFORMÁTICA	
3 33333	2 1997	SUP	LIC INFORMÁTICA	

Salida de DBMS

```

DNI: 12345
Universidad: 1
Año: 1992
Grado: MED
Especialidad: PRUEBA

DNI: 12345
Universidad: 1
Año: 1992
Grado: MED
Especialidad: PRUEBA

DNI: 12345
Universidad: 1
Año: 1992
Grado: MED
Especialidad: PRUEBAYUJYU

```

Comandos:

```

DECLARE
  v_DNI NUMBER := 12345;
  v_NOMBRE_UNIV VARCHAR2(25) := 'UNED';
  v_AGNO NUMBER := 1992;
  v_GRADO VARCHAR2(5) := 'MED';
  v_ESPECIALIDAD VARCHAR2(20) := 'PRUEBAYUJYU';
BEGIN
  -- Llamamos al procedimiento para actualizar los estudios

```



```

UPDATE Empleados
SET DIRECC1 = 'C/Saturno',
    CIUDAD = 'Las Palmas',
    SALARIO = SALARIO + 1000
WHERE DNI = 12345;

UPDATE Empleados
SET DIRECC1 = 'C/La tortuga',
    CIUDAD = 'Santa Cruz',
    SALARIO = SALARIO + 400
WHERE DNI = 33333;

select * from EMPLEADOS;
select * from CAMBIOSEMPLEADOS;

```

DNI	NOMBRE	APELLIDO1	APELLIDO2	SALARIO	DIRECC1	DIRECC2	CIUDAD	MUNICIPIO	COD_POSTAL	SEXO	FECHA_NAC
1	12345 Jose	Merce	LÃpez	3500	C/Saturno	C/ Otra, 1	Las Palmas	CÃdiz	11000	H	05/01/0074
2	22222 MarÃa	Rosal	CÃsar	2000	(null)	(null)	Ubrique	CÃdiz	11600	M	(null)
3	33333 Pilar	PÃrez	RollÃn	1400	C/La tortuga	(null)	Santa Cruz	(null)	11600	M	02/08/0073

Comandos:

```

CREATE TABLE CAMBIOSEMPLEADOS (
    CambioID NUMBER PRIMARY KEY,
    EmpleadoID NUMBER,
    NOMBRE_ANTIGUO VARCHAR2(50),
    APELLIDO1_ANTIGUO VARCHAR2(50),
    APELLIDO2_ANTIGUO VARCHAR2(50),
    SALARIO_ANTIGUO NUMBER(10,2),
    DIRECC1_ANTIGUO VARCHAR2(100),
    DIRECC2_ANTIGUO VARCHAR2(100),
    CIUDAD_ANTIGUO VARCHAR2(50),
    MUNICIPIO_ANTIGUO VARCHAR2(50),
    COD_POSTAL_ANTIGUO VARCHAR2(10),
    SEXO_ANTIGUO CHAR(1),
    FECHA_NAC_ANTIGUA DATE,
    FECHA_MODIFICACION DATE
);

```

b) Crear un trigger que se dispare cada vez que se haga una actualización de los datos de un empleado (a excepción del salario), se deben copiar los datos antiguos a la tabla CambiosEmpleados.

Comandos:

```
CREATE OR REPLACE TRIGGER Trg_GuardarCambiosEmpleados
BEFORE UPDATE ON EMPLEADOS
FOR EACH ROW
BEGIN
    INSERT INTO CAMBIOSEMPLEADOS (
        CambioID,
        EmpleadoID,
        NOMBRE_ANTIGUO,
        APELLIDO1_ANTIGUO,
        APELLIDO2_ANTIGUO,
        SALARIO_ANTIGUO,
        DIRECC1_ANTIGUO,
        DIRECC2_ANTIGUO,
        CIUDAD_ANTIGUO,
        MUNICIPIO_ANTIGUO,
        COD_POSTAL_ANTIGUO,
        SEXO_ANTIGUO,
        FECHA_NAC_ANTIGUA,
        FECHA_MODIFICACION
    ) VALUES (
        CambioID_Seq.NEXTVAL,
        :OLD.DNI,
        :OLD.NOMBRE,
        :OLD.APELLIDO1,
        :OLD.APELLIDO2,
        :OLD.SALARIO,
        :OLD.DIRECC1,
        :OLD.DIRECC2,
        :OLD.CIUDAD,
        :OLD.MUNICIPIO,
        :OLD.COD_POSTAL,
        :OLD.SEXO,
        :OLD.FECHA_NAC,
        SYSDATE
    );
END;
/
```

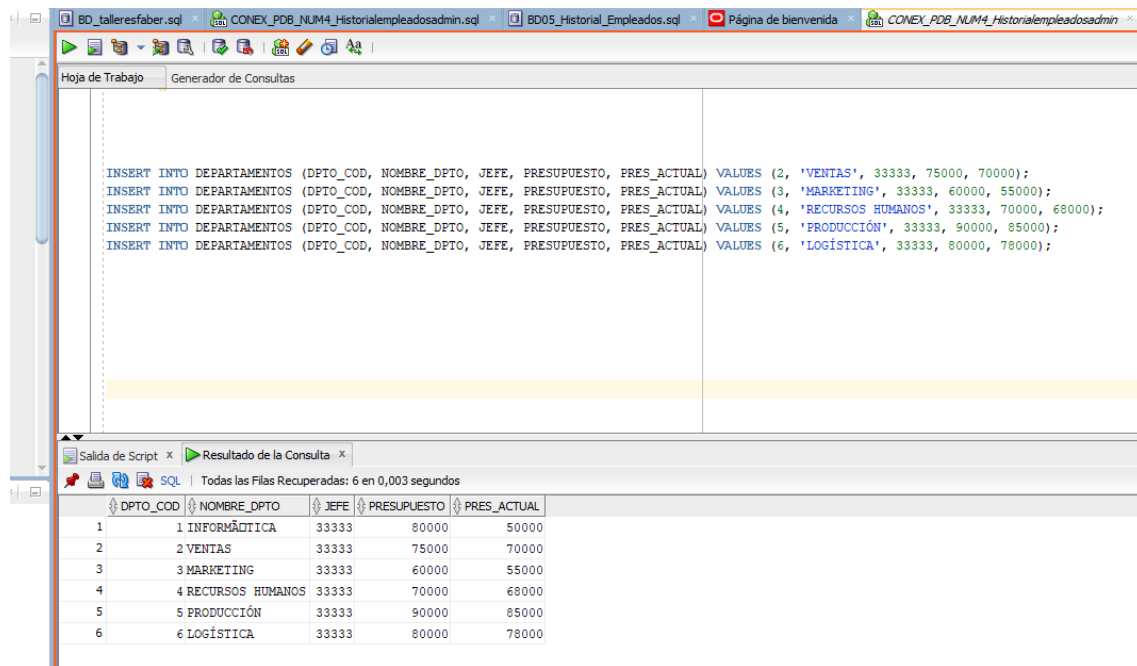
c) Realiza la comprobación.

Comandos:

```
UPDATE EMPLEADOS
SET DIRECC1 = 'Nueva dirección',
    CIUDAD = 'Nueva ciudad',
    SALARIO = SALARIO + 1000
WHERE DNI = 12345;
```


8.- Trigger y procedimiento:

a) Crear un procedimiento para cambiar a un empleado de departamento (pasados como parámetros el DNI del empleado, código del puesto de trabajo, fecha de inicio, el nombre del departamento y el dni del supervisor).



Antes de empezar habrá que crear nuevos departamentos en la tabla DEPARTAMENTOS, en concreto con el del ejemplo sería suficiente, el departamento de VENTAS, pero he creado algunos mas:

Comandos:

```
INSERT INTO DEPARTAMENTOS (DPTO_COD, NOMBRE_DPTO, JEFE, PRESUPUESTO, PRES_ACTUAL) VALUES (2, 'VENTAS', 33333, 75000, 70000);
INSERT INTO DEPARTAMENTOS (DPTO_COD, NOMBRE_DPTO, JEFE, PRESUPUESTO, PRES_ACTUAL) VALUES (3, 'MARKETING', 33333, 60000, 55000);
INSERT INTO DEPARTAMENTOS (DPTO_COD, NOMBRE_DPTO, JEFE, PRESUPUESTO, PRES_ACTUAL) VALUES (4, 'RECURSOS HUMANOS', 33333, 70000, 68000);
INSERT INTO DEPARTAMENTOS (DPTO_COD, NOMBRE_DPTO, JEFE, PRESUPUESTO, PRES_ACTUAL) VALUES (5, 'PRODUCCIÓN', 33333, 90000, 85000);
INSERT INTO DEPARTAMENTOS (DPTO_COD, NOMBRE_DPTO, JEFE, PRESUPUESTO, PRES_ACTUAL) VALUES (6, 'LOGÍSTICA', 33333, 80000, 78000);
```

El procedimiento:

BD_talleresfaber.sql CONEX_PDB_NUM4_Historiaempleadosadmin.sql BD05_Historial_Empleados.sql CONEX_PDB_NUM4_Historiaempleadosadmin1.sql Página de bienvenida

Hoja de Trabajo Generador de Consultas

```

SET FECHA_FIN = FECHA_INI
WHERE EMPLEADO_DNI = DNI AND FECHA_FIN IS NULL;
END IF;

SELECT DPTO_COD INTO COD_DEPT
FROM DEPARTAMENTOS
WHERE NOMBRE_DPTO = NOM_DEPT;

INSERT INTO HISTORIAL_LABORAL
VALUES (DNI, COD_PUESTO, FECHA_INI, NULL, COD_DEPT, DNI_SU);

COMMIT;
END;
/

BEGIN
CAMBIO_DEPARTAMENTO(12345, 1, TO_DATE('15-02-2023', 'DD-MM-YYYY'), 'VENTAS', 22222);
END;
/

CREATE OR REPLACE TRIGGER TRG_ACTUALIZARHISTORIALLABORAL
BEFORE INSERT ON HISTORIAL_LABORAL
FOR EACH ROW
BEGIN
-- VERIFICAMOS QUE EL NUEVO REGISTRO TIENE UNA FECHA DE INICIO POSTERIOR AL REGISTRO EXISTENTE
IF :NEW.FECHA_INICIO <= :OLD.FECHA_FIN THEN
-- SI LA FECHA DE INICIO NO ES POSTERIOR, ENTONCES NO SE PERMITE INSERTAR LA FECHA
RAISE_APPLICATION_ERROR(-20001, 'ERROR: La fecha de inicio debe ser posterior a la fecha de finalización en el historial laboral.');
```

Salida de Script x Resultado de la Consulta x

Todas las Filas Recuperadas: 16 en 0,006 segundos

	EMPLEADO_DNI	TRAB_COD	FECHA_INICIO	FECHA_FIN	DPTO_COD	SUPERVISOR_DNI
1	12345	1	05/01/03	01/01/18	1	33333
2	33333	4	15/01/01	(null)	1	33333
3	22222	3	03/11/04	03/11/05	1	33333
4	22222	3	03/11/05	(null)	1	33333
5	12345	1	01/01/18	10/05/20	2	22222
6	12345	1	10/05/20	10/05/22	2	22222
7	12345	1	10/05/22	10/08/22	2	22222
8	12345	1	10/08/22	10/10/22	2	22222
9	12345	1	10/10/22	10/11/22	2	22222
10	12345	1	10/11/22	15/11/22	2	22222
11	12345	1	15/11/22	18/11/22	2	22222
12	12345	1	18/11/22	18/12/22	2	22222
13	12345	1	18/12/22	10/01/23	2	22222
14	12345	1	10/01/23	10/02/23	2	22222
15	12345	1	10/02/23	15/02/23	2	22222
16	12345	1	15/02/23	(null)	2	22222

Comandos:

```

CREATE OR REPLACE PROCEDURE CAMBIO_DEPARTAMENTO(
  DNI NUMBER,
  COD_PUESTO NUMBER,
  FECHA_INI DATE,
  NOM_DEPT VARCHAR2,
  DNI_SU NUMBER
)
IS
  COD_DEPT NUMBER;
  DNI_BUSQ NUMBER;
BEGIN
  SELECT EMPLEADO_DNI INTO DNI_BUSQ
  FROM HISTORIAL_LABORAL
```

```

WHERE EMPLEADO_DNI = DNI AND FECHA_FIN IS NULL;

IF DNI_BUSQ IS NOT NULL THEN
    UPDATE HISTORIAL_LABORAL
    SET FECHA_FIN = FECHA_INI
    WHERE EMPLEADO_DNI = DNI AND FECHA_FIN IS NULL;
END IF;

SELECT DPTO_COD INTO COD_DEPT
FROM DEPARTAMENTOS
WHERE NOMBRE_DPTO = NOM_DEPT;

INSERT INTO HISTORIAL_LABORAL
VALUES (DNI, COD_PUESTO, FECHA_INI, NULL, COD_DEPT, DNI_SU);

COMMIT;
END;
/

```

Para probarlo introduciré un cambio de departamento en un empleado:

Comandos:

```

BEGIN
    CAMBIO_DEPARTAMENTO(12345, 1, TO_DATE('15-02-2023', 'DD-MM-YYYY'),
    'VENTAS', 22222);
END;
/

```

b) Crear un trigger para que cuando un empleado cambie de puesto actualice automáticamente su historial laboral (fecha de finalización en su anterior departamento). Realizar la comprobación con el procedimiento anterior.

Comandos:

```

CREATE OR REPLACE TRIGGER TRG_ACTUALIZARHISTORIALLABORAL
BEFORE INSERT ON HISTORIAL_LABORAL
FOR EACH ROW
BEGIN
    -- VERIFICAMOS QUE EL NUEVO REGISTRO TIENE UNA FECHA DE INICIO
    POSTERIOR AL REGISTRO EXISTENTE
    IF :NEW.FECHA_INICIO <= :OLD.FECHA_FIN THEN
        -- SI LA FECHA DE INICIO NO ES POSTERIOR, ENTONCES NO SE PERMITE
        INSERTAR LA FECHA
    END IF;
END;

```

```
RAISE_APPLICATION_ERROR(-20001, 'ERROR: La fecha de inicio debe  
ser posterior a la fecha de finalización en el historial laboral.');
```

```
END IF;  
END;  
/
```

FIN