Carga las tablas: PERSONAL, PROFESORES y CENTROS.

1. Crea un bloque anónimo que al ejecutarse presente todos los datos del centro con menos número de plazas.

   Si hubiese más de un centro con el número menor de plazas, aparecerá el mensaje “No se puede determinar el centro porque varios cumplen la condición”.

```
declare
  v_centro centros%rowtype;
begin
    select * into v_centro from centros
          where num_plazas=(select min(num_plazas) from centros);
    dbms_output.put_line('El centro con menos plazas es:');
    dbms_output.put_line(v_centro.cod_centro||' '||v_centro.tipo_centro||'
           '||v_centro.nombre||' '||v_centro.direccion||'
           '||v_centro.telefono||' '||v_centro.num_plazas);
exception
    when too_many_rows then
        dbms_output.put_line(' No se puede determinar el centro porque varios
          cumplen la condición');
end;
/
```

2. Modifica el bloque anterior para que se pregunte por teclado el número de plazas. El programa buscará el centro con esas plazas. Si no lo encuentra, aparecerá el mensaje “Ningún centro cumple la condición”.

```
declare
  v_centro centros%rowtype;
begin
    select * into v_centro from centros
          where num_plazas=&usuario_numplazas;
    dbms_output.put_line('El centro con menos plazas es:');
    dbms_output.put_line(v_centro.cod_centro||' '||v_centro.tipo_centro||'
          '||v_centro.nombre||' '||v_centro.direccion||'
          '||v_centro.telefono||' '||v_centro.num_plazas);
exception
    when too_many_rows then
                dbms_output.put_line(' No se puede determinar el centro
          porque varios cumplen la condición');
    when no_data_found then
       dbms_output.put_line('Ningún centro cumple la condición');
end;/
```

3. Crea un procedimiento llamado MAS_PLAZAS que aumente las plazas de los centros de un determinado tipo. Al procedimiento se le pasarán dos argumentos (parámetros) : código de centro y número de plazas a aumentar. Si el centro no existe dará un mensaje indicándolo.

```
create or replace
 procedure mas_plazas(v_codcentro centros.cod_centro%type, masplazas number)
as
begin
   update centros set num_plazas=num_plazas+masplazas
          where cod_centro=v_codcentro;
   if (sql%rowcount = 0) then
          dbms_output.put_line('No se ha encontrado el centro');
   end if;
end mas_plazas;
/
```

4. Crea un procedimiento llamado SALARIO_CONSERJES que aumente o disminuya el salario a los conserjes. Al procedimiento se le pasarán dos argumentos (parámetros): cantidad y tipo de operación ('A' aumentar, 'D' disminuir'). Utilizar CASE.
   Nota: si el segundo parámetro es distinto de 'A' o 'D' aparecerá el mensaje:

```
Opción incorrecta, utilice "A" para aumentar y "D" para disminuir
```

```
create or replace
procedure salario_conserjes(cantidad number, tipo char)
as
    v_cantidad number;
begin
   case tipo
     when 'A' then
      v_cantidad:=cantidad;
     when 'D' then
          v_cantidad:=cantidad*(-1);
     else
        dbms_output.put_line('Opción incorrecta, utilice "A" para aumentar y
                                    "D" para disminuir');
          v_cantidad:=0;
     end case;
update personal set salario=salario+v_cantidad
          where funcion='CONSERJE';
end salario_conserjes;
/
```

5. Dado el DNI de un profesor, crea un procedimiento BORRA_PROFESOR que borre ese profesor de la base de datos (de todas las tablas en que aparezca), mediante el mínimo número de comandos SQL.

   Como resultado se indicará con un mensaje si se ha podido borrar o no el profesor.

```
create or replace
procedure borra_profesor (dniBorrar personal.dni%type)
as
begin
 delete from personal where dni=dniBorrar; --como es borrado en cascada, se
                      borrará también de tabla profesores
 if (sql%rowcount=0) then
    dbms_output.put_line('No se ha encontrado ese DNI de profesor');
 else
    dbms_output.put_line('Profesor borrado');
 end if;
end borra_profesor;
/
```

6. Dado el código de un centro, crea un procedimiento BORRA_CENTRO que borre ese centro de la base de datos (de todas las tablas en que aparezca), mediante el mínimo número de comandos SQL.

   Como resultado se indicará con un mensaje si se ha podido borrar o no el centro. No se emplearán excepciones.

```
create or replace
procedure borra_centro (centroBorrar centros.cod_centro%type)
as
begin
   -- es borrado restringido, primero hay que borrarlo de profesores
```

```
    delete from profesores where cod_centro=centroBorrar;
    delete from centros where cod_centro=centroBorrar;
    if (sql%rowcount=0) then
       dbms_output.put_line('No se ha encontrado ese código de centro');
    else
       dbms_output.put_line('centro borrado');
    end if;
end borra_centro;
/
```

7. Escribe una función ESP_PROFESOR que, dado un DNI de profesor, devuelva el número de especialidades distintas que imparte (por ejemplo el profesor con DNI 9800990 imparte 1 especialidad, mientras que el que tiene DNI 1112346 imparte 2 especialidades).

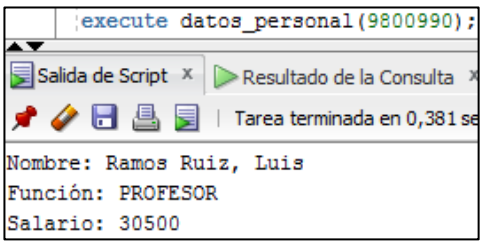
```
create or replace
function esp_profesor(v_dni profesores.dni%type)
return number
as
    resultado number;
begin
    select count(distinct especialidad) into resultado
       from profesores
       where dni=v_dni;
    return (resultado);
end esp_profesor;
/
```

8. Escribe un procedimiento CENTROS_VACIOS que disminuya un 20% las plazas de los centros sin profesores.

   Como resultado se indicará el número de centros modificados.

```
create or replace
procedure centros_vacios
as
begin
update centros set num_plazas=num_plazas*0.8 where cod_centro not in (select
cod_centro from profesores);
dbms_output.put_line('Se han actualizado '||sql%rowcount||' centros');
end centros_vacios;
/
```

9. Escribe un procedimiento DATOS_PERSONAL que, dado un DNI, devuelva sus datos de la tabla PERSONAL, de la forma:

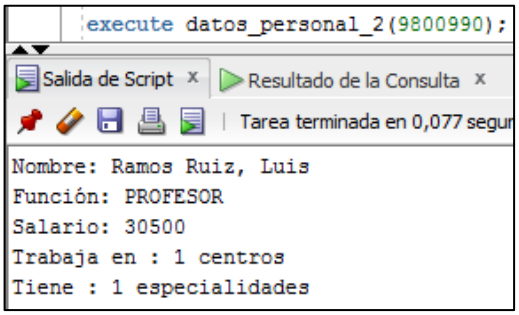


   Si no se encuentra el dni se mostrará un mensaje indicándolo.

```
create or replace
procedure datos_personal (v_dni personal.dni%type)
as
    datos personal%rowtype;
begin
    select * into datos from personal where dni=v_dni;
    dbms_output.put_line('Nombre: '||datos.apellidos);
    dbms_output.put_line('Función: '||datos.funcion);
    dbms_output.put_line('Salario: '||datos.salario);
exception
   when no_data_found then
         dbms_output.put_line('No se ha encontrado al trabajador');
end datos_personal;
/
```

10. Modifica DATOS_PERSONAL, llamándolo ahora DATOS_PERSONAL_2, de forma que si la persona es un profesor, aparezca, además de los datos anteriores, también el número de centros en los que trabaja y el número de espacialidades que tiene.

Ejemplo de ejecución:



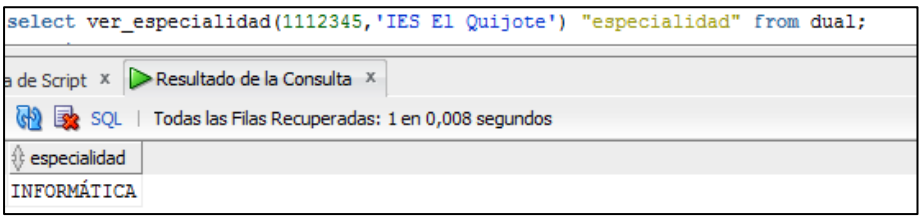

```
create or replace
procedure datos_personal_2 (v_dni personal.dni%type)
as
    v_datos personal%rowtype;
    v_numero_centros number;
    v_numero_especialidades number;
begin
    select * into v_datos from personal where dni=v_dni;
    dbms_output.put_line('Nombre: '||v_datos.apellidos);
    dbms_output.put_line('Función: '||v_datos.funcion);
    dbms_output.put_line('Salario: '||v_datos.salario);
    if (v_datos.funcion='PROFESOR') then
       select count(*) into v_numero_centros from profesores
                       where dni=v_dni;
       select count(distinct especialidad) into v_numero_especialidades from
                      profesores where dni=v_dni;
       dbms_output.put_line('Trabaja en : '||v_numero_centros||      ' centros');
        dbms_output.put_line('Tiene : '||v_numero_especialidades || ' especialidades');
     end if;
exception
   when no_data_found then
         dbms_output.put_line('No se ha encontrado al trabajador');
end datos_personal_2;
/
```

11. Escribe una función MAYOR_CENTRO que devuelva el nombre del centro con más plazas.

```
create or replace
function mayor_centro
return centros.nombre%type
as
   v_nombre centros.nombre%type;
begin
   select nombre into v_nombre from centros
          where num_plazas=(select max(num_plazas) from centros);
   return (v_nombre);
end mayor_centro;
/
```

12. Escribe una función VER_ESPECIALIDAD que devuelva la especialidad de un profesor, dados como argumentos su dni y el nombre del centro.

Ejemplo de ejecución:




```
create or replace
function ver_especialidad (v_dni profesores.dni%type, v_nombre_centro
                           centros.nombre%type)
return profesores.especialidad%type
as
   v_especialidad profesores.especialidad%type;
begin
   select especialidad into v_especialidad from profesores
      where dni=v_dni and cod_centro=
         (select cod_centro from centros
           where nombre=v_nombre_centro);

--otra forma:
--select especialidad into v_especialidad
--from profesores P, centros C
--where P.cod_centro=C.cod_centro and P.dni=v_dni and
--C.nombre=v_nombre_centro;

return (v_especialidad);
end ver_especialidad;
/
```

13. Escribe una función MENOR_CENTRO_xTIPO a la que se pasará como argumento el tipo de centro, y devolverá el nombre del centro de ese tipo con menos plazas.

```
create or replace
function menor_centro_xtipo (v_tipo_centro centros.tipo_centro%type)
return centros.nombre%type
as
   v_nombre centros.nombre%type;
begin
   select nombre into v_nombre from centros
          where tipo_centro=v_tipo_centro and
                num_plazas=(select min(num_plazas) from centros
                where tipo_centro=v_tipo_centro);
   return (v_nombre);
end menor_centro_xtipo;
/
```