1. **La base de datos Academia permite gestionar las inscripciones de los alumnos a los diferentes cursos que ofrecen.**

   a. Crear una función que devuelva un valor boolean que indique si hay cupo en un curso para una nueva inscripción.
   b. Crear un procedimiento para realizar una inscripción en un determinado curso en caso de que haya cupo; en otro caso se mostrará un mensaje indicando la imposibilidad de realizarla. El procedimiento recibirá 3 parámetros: código del curso, identificador del estudiante, y un valor boolean que indique si es o no antiguo alumno. El importe de la inscripción corresponderá al indicado en las cuotas en el caso de inscripción en cursos regulares o intensivos. La cuota de los cursos particulares corresponde a una hora de clase, por lo que la inscripción dependerá del número de horas asociado al curso. A los antiguos alumnos se les aplicará un descuento del 5% en todos los cursos.
   c. Crear un procedimiento que liste para cada curso (código, nombre y nivel) el número de inscripciones y el importe total recaudado, ordenado por nivel. Parametrizar el procedimiento para poder indicar que se muestren solo cursos con un número mínimo de matrículas.

2. **Crea las tablas Contratos y Trayectos.**

```
CREATE TABLE Contratos(Referencia   VARCHAR(10) PRIMARY KEY,
                       Empresa      VARCHAR(100),
                       Fecha        DATE,
                       NumTrayectos NUMBER(2,0));


CREATE TABLE Trayectos(Referencia VARCHAR(10) REFERENCES Contratos ON
DELETE CASCADE,
                     Origen     VARCHAR(50),
                     Destino    VARCHAR(50),
                     Vehículo   VARCHAR(20),
                     PRIMARY KEY (Referencia, Origen, Destino));
```

   a. Escribir un procedimiento que reciba una referencia de contrato (que se asume que existe) como parámetro de entrada y actualice su contador de trayectos (NumTrayectos) y lo imprima por consola. Se debe declarar una excepción que se lance para dar un mensaje si la referencia no tiene trayectos.
   b. Crear un trigger que mantenga actualizado el atributo redundante al insertar y borrar en la tabla (se asume que antes de la actualización este atributo es consistente).

3. **Crea las tablas Empleados y Registro de cambios.**

```
CREATE TABLE Empleados( DNI    CHAR(9) PRIMARY KEY,
                        Nombre      VARCHAR(100),
                        CodDept     CHAR(5) REFERENCES Departamentos on
                  delete set NULL,
                        Salario     NUMBER(4,0));

CREATE TABLE Departamentos(CodDept   CHAR(5) PRIMARY KEY,
                           Nombre      VARCHAR(100));
```

```
CREATE TABLE Cambios(IdCambio VARCHAR(10) PRIMARY KEY,
                     Usuario VARCHAR(8),
                      SalarioAnt  NUMBER(4,0),
                       SalarioNew  NUMBER(4,0));
```

a. Implementar un trigger que registre en la tabla Cambios cualquier modificación que se produzca en el salario de un empleado, indicando el usuario y la fecha en la que se realizó. El identificador se obtendrá de una secuencia denominada SEQCambios.
b. Escribir un procedimiento almacenado que liste por departamento el nombre y salario de cada empleado cuyo salario sea inferior a la media del departamento. Incluir el total de dichos salarios por departamento

4. **Ejecutar las siguientes instrucciones:**

```
drop table ComisionCC;
drop table deposito;
drop table log;
create table ComisionCC(cc char(20), importe number(10,2));
create table deposito(cc char(20));
create table log( msg varchar(50));
```

Diseñar un trigger asociado a la operación delete de la tabla ComisionCC, de modo que si la cuenta del registro que se borre se encuentra en la tabla deposito indique en log un mensaje que indique la cc, el importe y el texto “Deposito asociado”. En caso contrario el texto indicará “Cliente preferente”

Hacer las siguientes pruebas para comprobar el funcionamiento:

```
insert into Comisioncc values ('12345678900987654321',13.9);
insert into Comisioncc values('12345123131333344321',13.0);
insert into Comisioncc values ('37423462487654321478',13.9);
insert into deposito values ('37423462487654321478');
delete from ComisionCC;
```

5. **Ejecutar las siguientes instrucciones:**

```
drop table Records;
drop table Marcas;
create table Records(prueba number  primary key,  tiempo number);
create table Marcas(prueba number, fecha date, tiempo number, primary key
(prueba,fecha));
```

Diseñar un trigger asociado a la operación de inserción de la tabla Marcas, de modo que si el tiempo de la prueba que se inserte es un nuevo record se actualice el registro correspondiente en la tabla Records.

Realiza las siguientes pruebas

```
delete from Marcas;
delete from Records;
```

```
insert into Marcas values (1, to_date('01/02/2013'),3.8);
insert into Marcas values (1, to_date('02/02/2013'),4.2);
insert into Marcas values (1, to_date('03/02/2013'),3.5);
```

6. **Ejecutar las siguientes instrucciones:**

```
drop table Libros cascade constraints;
drop table Ejemplares cascade constraints;
create table Libros(isbn char(13) primary key,
                    copias integer);
create table Ejemplares(signatura char(5) primary key,
                        isbn char(13) not null,
                        FOREIGN KEY (isbn)
                        REFERENCES Libros);
```

Escribir un trigger asociado a la inserción de filas en Ejemplares, de forma que si el isbn no aparece en Libros, se cree una fila en Libros con dicho isbn y copias con valor 1, de forma que se evite el error por la violación de la foreign key. En caso de existir, el número de ejemplares se incrementará en uno. Prueba insertando Ejemplares que satisfagan ambas condiciones.