1. **Execute the script ScriptExercises PLSQL.sql. A procedure called “LOADEVENTS” will be created. You must execute it in order to insert rows in the different tables.**
2. Write a function with a course code as input parameter. It must return a Boolean value. The function will return TRUE if the number of students registered in the course is less than the quota (column quota in table courses) that can be accepted for this specific course. Otherwise, the function will return FALSE.
3. Write a stored procedure to enroll a student in a given course. First, you must check that the quota has not been exceeded (invoke the function previously designed); If the student cannot be enrolled, show a message that indicates it. The procedure will receive 3 parameters: course code, student id and a boolean that indicates if the student was enrolled previously in other course. In the case of regular and intensive courses the total price of the course will be the one stored in the table Fees. The price associated with the private courses is the price of one hour. In order to calculate the final price, you have to multiply it by the number of hours of each course. All the students that were enrolled previously in a course will obtain a 5% discount.
4. Write a stored procedure that lists for each course (course code, course name and level) the number of enrolled students and the total amount collected ordered by level. The procedure will receive a parameter that will be used to filter the courses that have been reached a minimum number of students.
5. **Create the following tables**

CREATE TABLE Contracts(Ref VARCHAR(10) PRIMARY KEY,

Organization VARCHAR(100),

ContDate DATE,

NumRoutes NUMBER(2,0));

CREATE TABLE Routes(Ref VARCHAR(10) REFERENCES Contracts ON DELETE CASCADE,

Origin VARCHAR(50),

Destination VARCHAR(50),

Vehicle VARCHAR(20),

PRIMARY KEY (Ref, Origin, Destination));

1. Write a stored procedure with a reference contract as input parameter. The procedure must update the information in NumRoutes, according to the number of routes associated with the contract. The procedure also must print the name of the organization and the total number of associated routes. You must declare an exception that is thrown to show a message if the reference does not have associated any route.
2. Create a trigger to keep updated the value of NumRoutes whenever a row is inserted or deleted.
3. **Create the following tables.**

CREATE TABLE Departments (CodDept CHAR(5) PRIMARY KEY,

Name VARCHAR(100));

CREATE TABLE Employees (SSN CHAR(9) PRIMARY KEY,

Name VARCHAR(100),

CodDept CHAR(5) REFERENCES Departments on delete set NULL,

Salary NUMBER(4,0));

CREATE TABLE Changes(IdChange VARCHAR(10) PRIMARY KEY,

UserId VARCHAR(8),

OldSalary NUMBER(4,0),

NewSalary NUMBER(4,0));

1. Write a trigger that records in the table Changes any update of the salary of the employees. The trigger must store the user, the date of the change, and both the salary before the change and the updated salary. The ID will be obtained from a sequence called SEQChanges.
2. Write a stored procedure that lists for each department the name and salary of each employee whose salary is lower than the average of the department. For each department the procedure must show the total amount of these salaries by department.
3. **Execute following statements.**

**drop table ComissionAC;**

**drop table deposit;**

**drop table log;**

**create table CommissionAC(acNumber char(20), amount number(10,2));**

**create table deposit (acNumber char(20));**

**create table log( msg varchar(50));**

Write a trigger that registers in the table log a message that includes the account number, the associated amount and the text “Associated Deposit” when a row from CommissionAC is deleted and the account is also stored in the table deposit. If the row is not stored in this table the message will include the text “Preferred Client”

Hacer las siguientes pruebas para comprobar el funcionamiento:

**insert into CommissionAC values ('12345678900987654321',13.9);**

**insert into CommissionAC values('12345123131333344321',13.0);**

**insert into CommissionAC values ('37423462487654321478',13.9);**

**insert into deposit values ('37423462487654321478');**

**delete from CommissionAC;**

1. **Execute following statements.**

**drop table Records;  
drop table Marks;  
create table Records(event number primary key, time number);  
create table Marks(event number, dateEvent date, tiempo number, primary key (event, dateEvent));**

Write a trigger associated to the insertion of a new row in the table Marks. If the new time is less than the record associated with the event , then this will update in the table Records.

Test the trigger executing these statements:

**delete from Marks;**

**delete from Records;**

**insert into Marks values (1, to\_date(‘01/02/2015’),3.8);**

**insert into Marks values (1, to\_date(‘02/02/2015’),4.2);**

**insert into Marks values (1, to\_date(‘03/02/2015’),3.5);**

1. **Execute following statements.**

**drop table Books cascade constraints;**

**drop table Copies cascade constraints;**

**create table Books(isbn char(13) primary key,**

**copiesNumber integer);**

**create table Copies(catalogNumber char(5) primary key,**

**isbn char(13) not null,**

**FOREIGN KEY (isbn) REFERENCES Books);**

Write a trigger associated to the insertion of a new row in the table Copies, such that, if the new isbn is not stored in the table Books, then a new row will be inserted in this table with the new isbn and copiesNumber equal to 1. If the isbn is stored in the table Books, the trigger will update the value of the column copiesNumber accordingly.

Test the trigger executing these statements.

**insert into Copies values (‘A1’,‘1234567891011’);**

**insert into Copies values (‘A2’,‘1234567891011’);**