

# Databases

## Practice #3 - Oracle SQLPLUS

This practice comprises two parts. Part I proposes some more queries on an existing database, to be solved using the SQL language. In Part II, you are requested to write the SQL scripts for creating and populating a database, whose logical schema is provided.

### PART I

#### Purpose

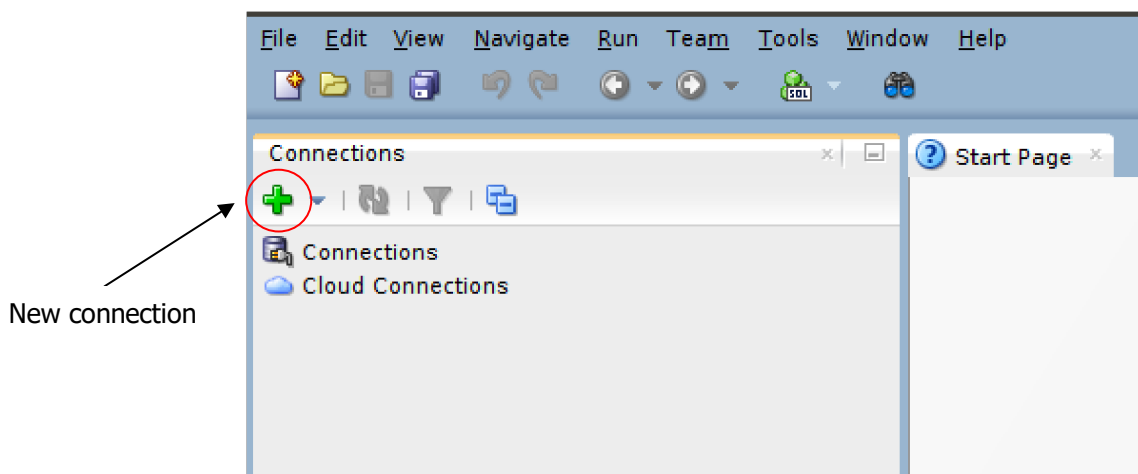
Write some SQL queries on an Oracle database.

#### Connection to the Oracle database

The SQL queries are executed through the SQL Developer software.

##### 1) Connection to the database

- Launch Oracle SQL Developer
- Click on New connection

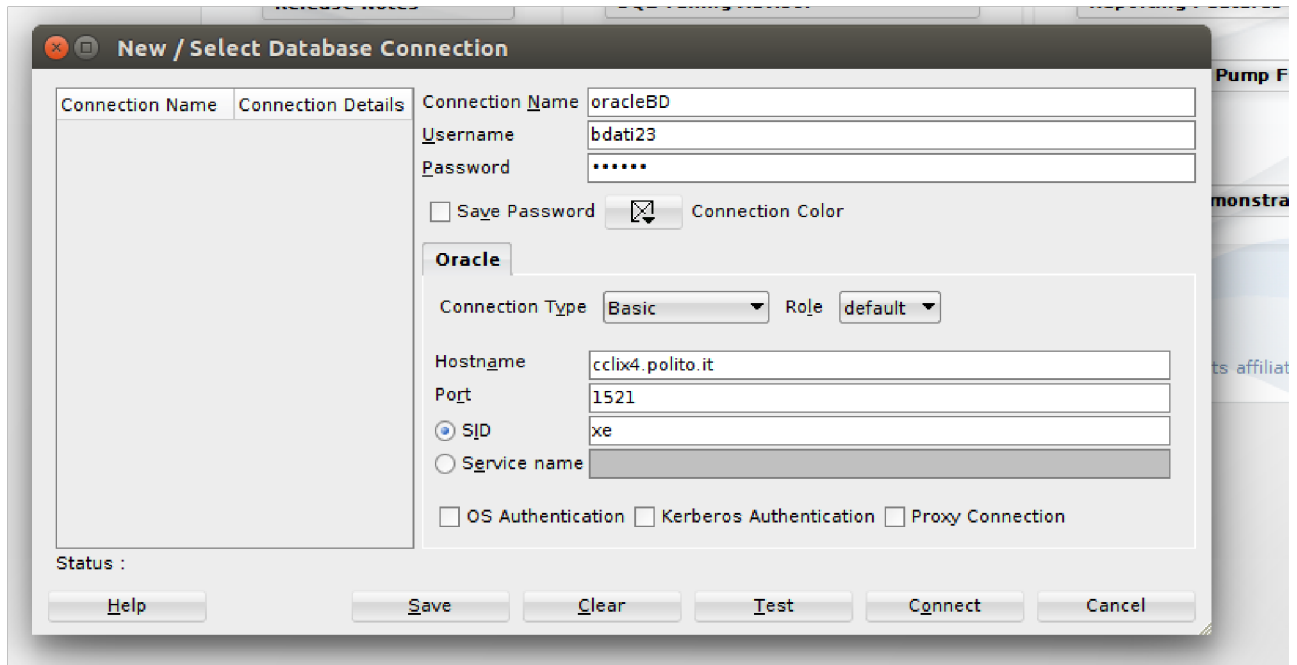


##### 2) Login

To login, you have to insert the following parameters:

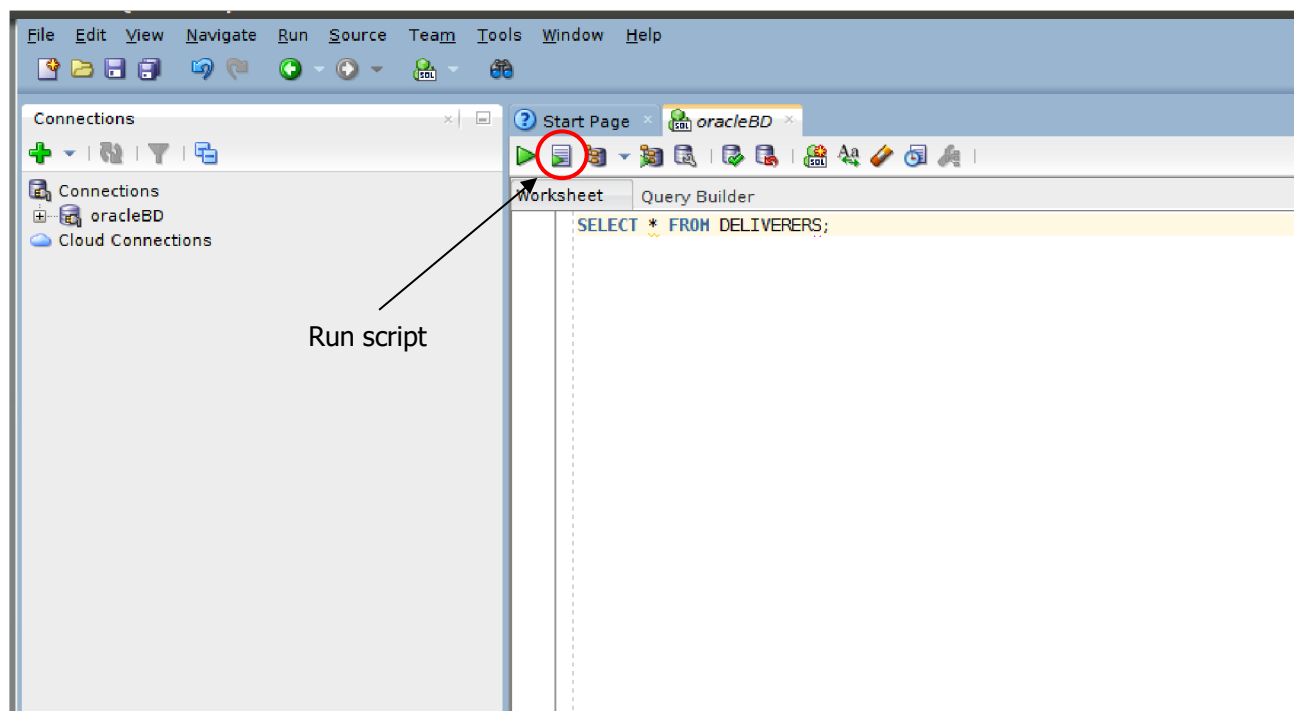
- Connection name: oracleBD
- Username: bdatiXY ○ XY are the last two digits of the PC number you are using
- Password: oracXY ○ XY are the last two digits of the PC number you are using
- Hostname: cclix4.polito.it
- Port: 1521
- SID: xe

For example, if you are working on pc number 23, the corresponding username is bdati15 and the password is orac23.



### **Write and execute SQL queries**

Write the SQL query in the Worksheet and execute it by clicking on the “Run script” button.



## 2. Description of the *Delivery* database

The Delivery database gathers information about the activities of a firm delivering and collecting goods for various customer companies.

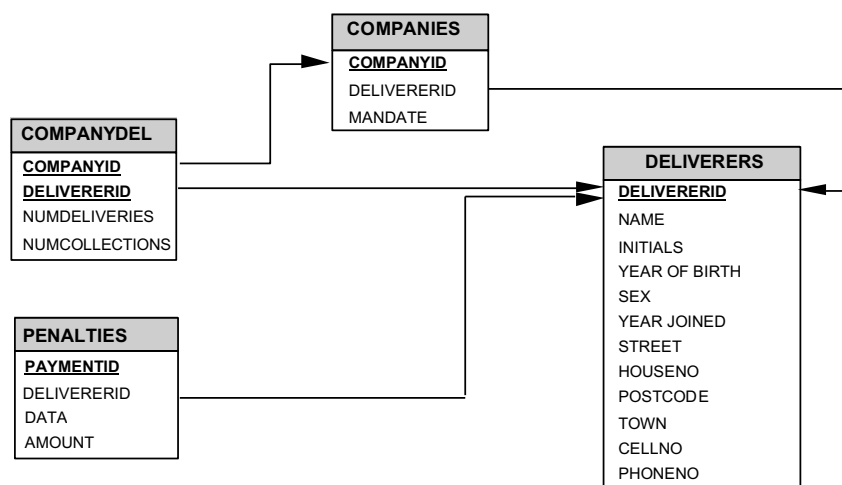
The DELIVERERS table contains the personal data for the deliverers working at the firm. For each deliverer, the following information is available: identification code (DELIVERERID), last name, first name initials, year of birth, sex, year when she/he began working for the firm, street, house number, city, residence postal code, cellular phone number, and office phone number.

The COMPANIES table reports, for each customer company, the company identification code (CompanyID) and the identification code of the deliverer who is the company's current reference person. In addition, it reports the number of times (MANDATE) the deliverer held this position.

The COMPANYDEL table reports the total number of good deliveries (NUMDELIVERIES) and collections (NUMCOLLECTIONS) made by each deliverer for each customer company. Note that the table only reports the deliverercompany pairs such that the deliverer performed at least one delivery or collection for the company.

The PENALTIES table reports the fines received by each deliverer. For each fine, the fine code (PENALTYID), the deliverer code, the fine date, and amount to be paid are stored.

The database schema is shown in the subsequent figure. Next Section 2 reports the table instance.



### 3. Table instance for the *Delivery* database

The primary key is underlined. Optional attributes are denoted with \*.

**DELIVERERS** table

<u>DELIVERERID</u>	NAME	INITIALS	YEAR_OF_BIRTH	SEX	YEAR JOINED	STREET	HOUSENO	POSTCODE	TOWN	CELLNO	PHONENO*
2	Everett	R	1948	M	1975	Stoney Road	43	3575NH	Stratford	070-237893	2411
6	Parmenter	R	1964	M	1977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963	M	1981	Edgecombe Way	39	9758VB	Stratford	070-347689	NULL
8	Newcastle	B	1962	F	1980	Station Road	4	6584WO	Inglewood	070-476573	2983
27	Collins	DD	1964	F	1983	Long Drive	804	8457DK	Eltham	079-234857	2513
28	Collins	C	1963	F	1983	Old main Road	10	1294QK	Midhurst	010-659599	NULL
39	Bishop	D	1956	M	1980	Eaton Square	78	9629CD	Stratford	070-393435	NULL
44	Baker	E	1963	M	1980	Lewis Street	23	4444LJ	Inglewood	070-368753	1124
57	Brown	M	1971	M	1985	Edgecombe Way	16	4377CB	Stratford	070-473458	6409
83	Hope	PK	1956	M	1982	Magdalene Road	16a	1812UP	Stratford	070-353548	1608
95	Miller	P	1934	M	1972	High Street	33a	5746OP	Douglas	070-867564	NULL
100	Parmenter	P	1963	M	1979	Haseltine Lane	80	1234KK	Stratford	070-476537	6524
104	Moorman	D	1970	F	1984	Stout Street	65	9437AO	Eltham	079-987571	7060
112	Bailey	IP	1963	F	1984	Vixen Road	8	6392LK	Plymouth	010-54874	1319

**COMPANYDEL** table

Tabella **PENALTIES**

<u>COMPANYID</u>	<u>DELIVERERID</u>	NUMD VERIF	<u>PAYMENTID</u>	DELIVERER ID	DATA	AMOUNT
1	2	4	1	6	12/08/1980	100
1	6	9	2	44	05/05/1981	75
1	8	0	3	27	10/09/1983	100
1	44	7	4	104	08/12/1984	50
1	57	5	5	44	08/12/1980	25
1	83	3	6	8	08/12/1980	25
2	8	4	7	44	30/12/1982	30
2	27	11	8	27	12/11/1984	75
2	104	8				
2	112	4				

**COMPANIES** table

<u>COMPANYID</u>	DELIVERERID	MANDATE
1	6	first
2	27	second

### 3. Queries

1. For each deliverer that has received at least two fines, find the identification code (of the deliverer), the date of the first fine and the date of the last fine.
2. For each deliverer that has been fined, find the identification code, the date of the last fine he/she received and the amount of this fine.
3. Find the identification codes of companies where more than 30% of the deliverers in the database have performed at least one delivery or one collection.

## PART II

### Purpose

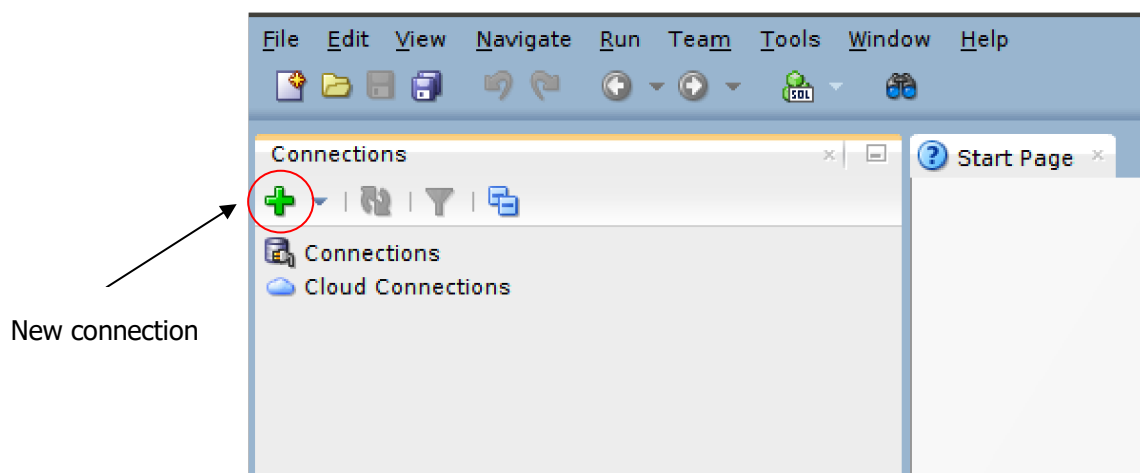
The purpose of this part is to write the scripts for creating and populating a database, given the logical schema, and to execute update and delete commands on this database, using the SQL language.

### Connection to the Oracle database

The SQL queries are executed through the SQL Developer software.

#### 3) Connection to the database

- Launch Oracle SQL Developer
- Click on New connection

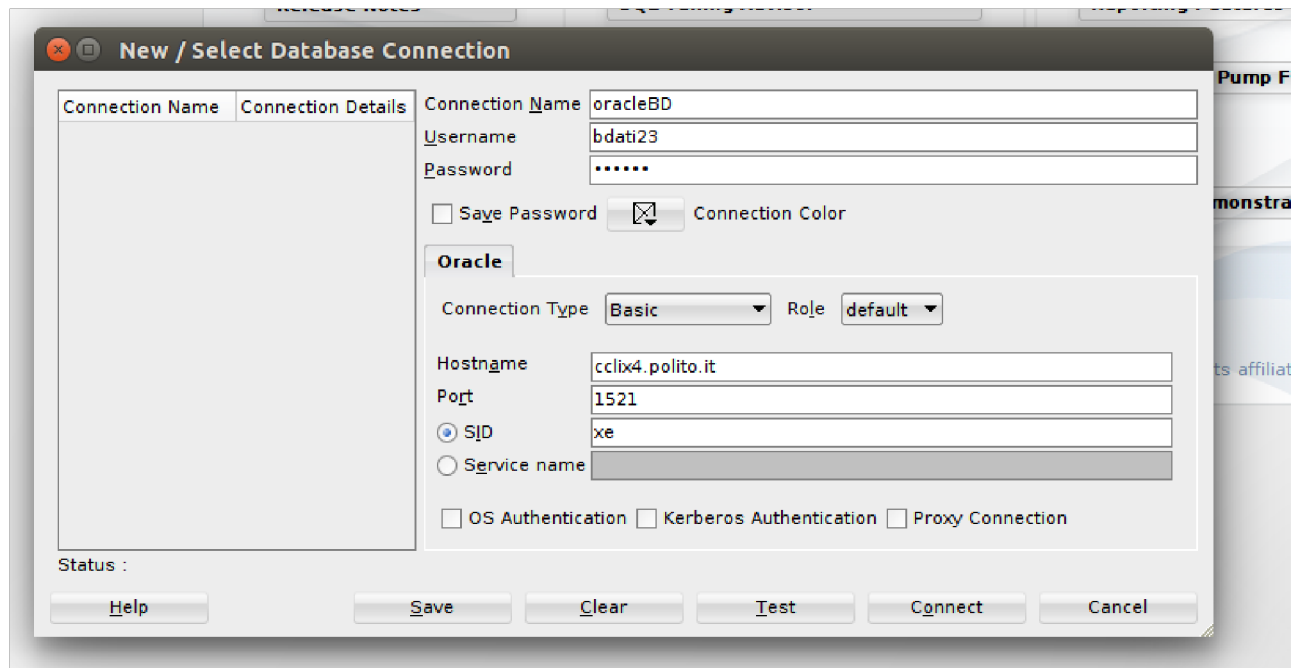


#### 4) **Login**

To logon, you have to insert the following parameters:

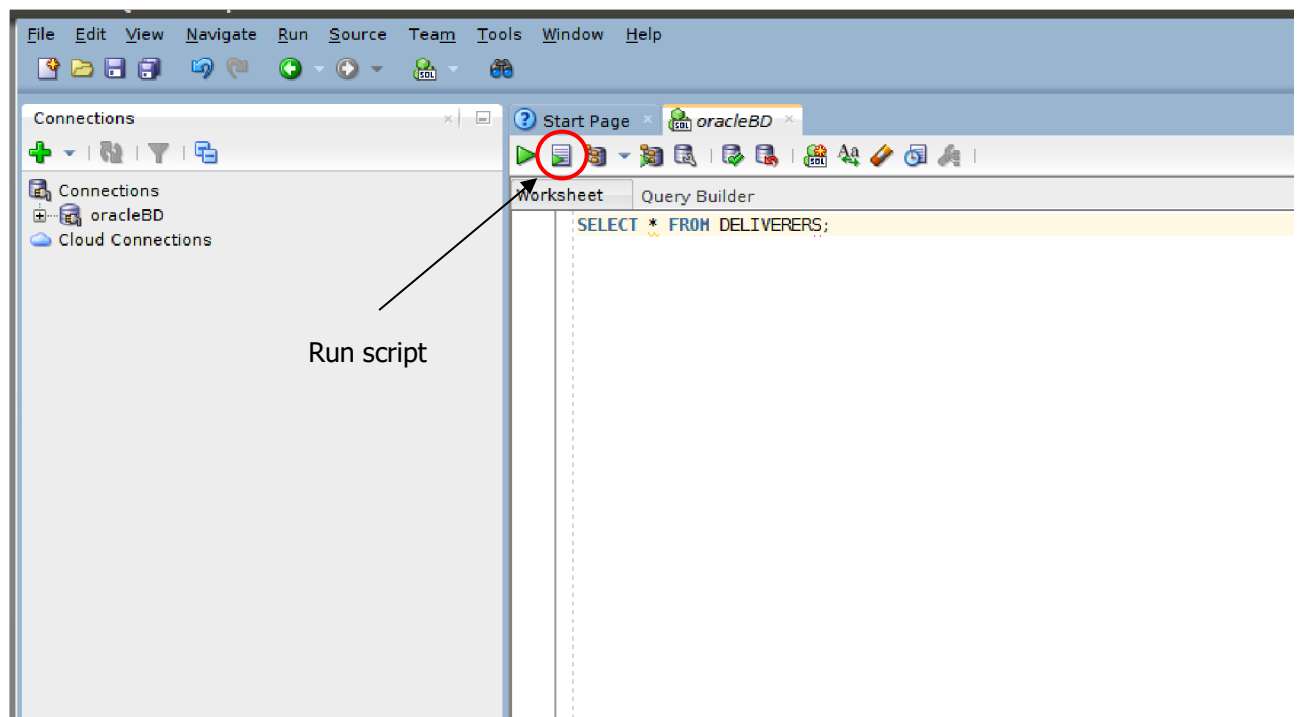
- Connection name: oracleBD
- Username: bdatiXY ○ XY are the last two digits of the PC number you are using
- Password: oracXY ○ XY are the last two digits of the PC number you are using
- Hostname: cclix4.polito.it
- Port: 1521
- SID: xe

For example, if you are working on pc number 23, the corresponding username is bdati15 and the password is orac23.



#### **Write and execute SQL scripts**

Write the script (set of SQL commands) in the Worksheet and execute it by clicking on the “Run script” button. Scripts can be loaded by using the open button.



## 1. Description of the *Gym* database

The database you should implement is about the activities in a gym. It is described by the following logical schema (primary keys are underlined, foreign keys are in italic, and optional attributes are denoted with \*):

TRAINER (SSN, Name, Surname, DateOfBirth, Email, PhoneNo\*)

COURSE (CId, Name, Type, Level)

SCHEDULE (SSN, Day, StartTime, Duration, *CId*, GymRoom)

For each trainer, the Social Security Number (SSN), the name, the surname, the date of birth, the email address, and the phone number (if any) are known. For each course, the code, the name, the type, and the level (1-4) are known. The course schedule lists the day of the week and the start time for each lesson of a given course taught by each trainer, together with the duration of the lesson (in minutes) and the gym room in which it is held.

TRAINER table

<u>SSN</u>	Name	Surname	DateOfBirth	Email	PhoneNo
SMTPLA80N31B791Z	Paul	Smith	31/12/1980	p.smith@gym.it	NULL
KHNJHN81E30C455Y	John	Johnson	30/5/1981	j.johnson@gym.it	+2300110303444
AAAGGG83E30C445A	Peter	Johnson	30/5/1981	p.johnson@gym.it	+2300110303444

COURSE table

<u>CId</u>	Name	Type	Level
------------	------	------	-------

CT100	Spinning for beginners	Spinning	1
CT101	Fitdancing	Music activity	2
CT104	Advanced spinning	Spinning	4

SCHEDULE table

<u>SSN</u>	<u>Day</u>	<u>StartTime</u>	<u>Duration</u>	<u>Cld</u>	<u>GymRoom</u>
SMTPLA80N31B791Z	Monday	10:00	45	CT100	R1
SMTPLA80N31B791Z	Tuesday	11:00	45	CT100	R1
SMTPLA80N31B791Z	Tuesday	15:00	45	CT100	R2
KHNJHN81E30C455Y	Monday	10:00	30	CT101	R2
KHNJHN81E30C455Y	Monday	11:30	30	CT104	R2
KHNJHN81E30C455Y	Wednesday	9:00	60	CT104	R1

Figure 1. Contents that the *Gym* database should include after executing the SQL script designed in this practice.

## 2. Scripts

1. Write an SQL script (*createDB.sql*) with the commands (i.e., CREATE TABLE) for creating the database corresponding to the logical schema described in Section 1.

In particular:

1. Define the three tables, choosing the most appropriate data type for each of the attributes. Pay special attention to the definition of primary keys and referential integrity constraints.
2. Choose the most appropriate constraint management policy in each context.

**Note.** Do pay attention to the order in which tables are created. Referenced tables should appear first in the script, while referencing tables must appear after the table(s) they reference.

2. Write an SQL script (*populateDB.sql*) containing the INSERT commands for populating the database created in the previous point. The script should include the insert commands required to obtain an instance of the database containing the same data shown in the tables in Figure 1.

**Note.** The order in which the insert commands are executed *does* matter. Make sure you follow the correct order so as not to violate the referential integrity constraints.

3. Test your *createDB.sql* and *populateDB.sql* scripts.

**Note.** The tables might already exist in the database if some other student has already created them using the same account. In this case, you should execute the following commands to delete them, prior to executing your own scripts:



1. DROP TABLE SCHEDULE;
  2. DROP TABLE COURSE;
  3. DROP TABLE TRAINER;
4. Write and execute the following update instructions in SQL, one by one, and check what happens in the database.
- 4.1. Update the phone number of trainer identified by SSN 'KHNJHN81E30C455Y', setting its value to '+390112333551'.
  - 4.2. Update the database in order to move in room 'R4' all the lessons scheduled in room 'R2'
  - 4.3. Remove from table COURSE all the courses scheduled once a week (i.e., appearing only once in table SCHEDULE). How is table COURSE affected by this command? And table SCHEDULE? The effect on the two tables is somehow related with the policy of constraints management selected during the creation of tables?
  - 4.4. Delete the trainer with SSN 'SMTPLA80N31B791Z'. How are tables TRAINER and SCHEDULE affected by this command? How is the result linked to the policy of constraints management selected during the creation of tables?

## Solution PART I

1. For each deliverer that has received at least two fines, find the identification code (of the deliverer), the date of the first fine and the date of the last fine.

```
SELECT DELIVEREDID, MIN(DATA), MAX(DATA)
```

## FROM PENALTIES

GROUP BY DELIVEREDID

HAVING COUNT(\*)>=2;

2. For each deliverer that has been fined, find the identification code, the date of the last fine he/she received and the amount of this fine.

SELECT P1.DELIVEREDID, DATA, AMOUNT

FROM PENALTIES P1

WHERE P1.DATA = (SELECT MAX(DATA)

FROM PENALTIES P2

WHERE P2.DELIVEREDID=P1.DELIVEREDID);

Find the identification codes of companies where more than 30% of the deliverers in the database have performed at least one delivery or one collection.

```
SELECT COMPANYID
FROM COMPANYDEL
GROUP BY COMPANYID
HAVING COUNT(*) > (SELECT 0.30*COUNT(*)
                    FROM DELIVERERS);
```

## Solution PART II

1. Write an SQL script (*createDB.sql*) with the commands (i.e., CREATE TABLE) for creating the database corresponding to the logical schema described in Section 1.

```
-- create an empty database. Name of the database:
SET storage_engine=InnoDB;
SET FOREIGN_KEY_CHECKS=1;
CREATE DATABASE IF NOT EXISTS gym;

-- use gym
use gym;

-- drop tables if they already exist
DROP TABLE IF EXISTS TRAINER;
DROP TABLE IF EXISTS COURSE;
DROP TABLE IF EXISTS SCHEDULE;

-- create tables

create table trainer (
    ssn char(20),
    name varchar(50) not null,
    surname varchar(50) not null,
    dateofbirth date not null,
    email varchar(100) not null,
    phoneno varchar(30) default null,
    primary key(ssn),
    constraint email_constr check(email like '%@%')
```

```

);

create table course (
    cid char(10),
    name varchar(50) not null,
    type varchar(50) not null,
    clevel smallint not null,
    primary key(cid),
    constraint lev_constr check(clevel >= 1 and clevel <=4)
);

create table schedule (
    ssn char(20),
    dayofweek char(9) not null,
    starttime time not null,
    duration smallint not null,
    cid char(10) not null,
    gymroom char(5) not null,
    primary key(ssn, dayofweek, starttime),
    foreign key(ssn) references trainer(ssn) on delete cascade on update cascade,
    foreign key(cid) references course(cid) on delete cascade on update cascade,
    constraint dayofweek_constr check(dayofweek in ('Monday', 'Tuesday',
'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'))
);

```

2. Write an SQL script (*populateDB.sql*) containing the INSERT commands for populating the database created in the previous point. The script should include the insert commands required to obtain an instance of the database containing the same data shown in the tables in Figure 1.

```

SET storage_engine=InnoDB;
SET FOREIGN_KEY_CHECKS=1;
use gym;

insert into trainer(ssn, name, surname, dateofbirth, email, phoneno)
values('SMTPLA80N31B791Z', 'Paul', 'Smith', to_date('31/12/1980', 'dd/mm/yyyy'),
'p.smith@gym.it', null);

insert into trainer(ssn, name, surname, dateofbirth, email, phoneno)
values('KHNJHN81E30C455Y', 'John', 'Johnson', to_date('30/05/1981',
'dd/mm/yyyy'), 'j.johnson@gym.it', '+2300110303444');

insert into trainer(ssn, name, surname, dateofbirth, email, phoneno)
values('AAAGGG83E30C445A', 'Peter', 'Johnson', to_date('30/05/1981',
'dd/mm/yyyy'), 'p.johnson@gym.it', '+2300110303444');

insert into course(cid, name, type, clevel) values('CT100', 'Spinning for beginners',
'Spinning', 1);

```

```
insert into course(cid, name, type, clevel) values('CT101', 'Fitdancing', 'Music activity', 2);
```

```
insert into course(cid, name, type, clevel) values('CT104', 'Advanced spinning', 'Spinning', 4);
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('SMTPLA80N31B791Z', 'Monday', '10:00', 45, 'CT100', 'R1');
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('SMTPLA80N31B791Z', 'Tuesday', '11:00', 45, 'CT100', 'R1');
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('SMTPLA80N31B791Z', 'Tuesday', '15:00', 45, 'CT100', 'R2');
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('KHNJHN81E30C455Y', 'Monday', '10:00', 30, 'CT101', 'R2');
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('KHNJHN81E30C455Y', 'Monday', '11:30', 30, 'CT101', 'R2');
```

```
insert into schedule(ssn, dayofweek, starttime, duration, cid, gymroom) values('KHNJHN81E30C455Y', 'Wednesday', '09:00', 30, 'CT101', 'R1');
```

3. Test your *createDB.sql* and *populateDB.sql* scripts.

4. Update/delete/insert check

4.1.UPDATE trainer

```
SET PhoneNo = '+390112333551'
WHERE SSN = 'KHNJHN81E30C455Y';
```

4.2.UPDATE schedule

```
SET GymRoom = 'S4'
WHERE GymRoom = 'S2';
```

4.3.DELETE FROM course

```
WHERE course.Cid IN (SELECT schedule.Cid
                     FROM schedule
                     GROUP BY schedule.Cid
                     HAVING COUNT(*)=1);
```

The row of table Course corresponding to the course with Cid CT101 is deleted. Also, thanks to the DELETE ON CASCADE option, the row corresponding to the same course in table Schedule is deleted.

#### 4.4.DELETE FROM trainer

```
WHERE SSN = 'SMTPLA80N31B791Z';
```

Thanks to the DELETE ON CASCADE option, also the rows of the table Schedule containing the SSN of the deleted trainer are canceled.