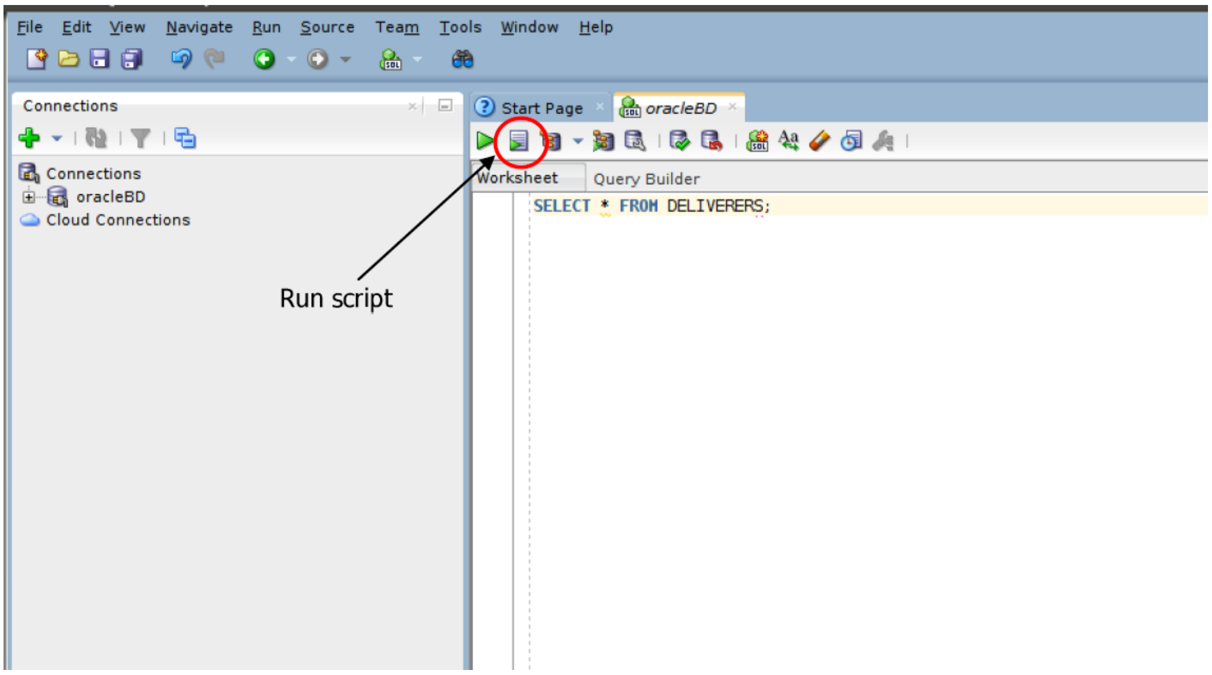
Introduction to Databases
Oracle SQLPLUS - Practice n. 3

This practice comprises two parts. Part I is based on Oracle DBMS and proposes some more queries on an existing database, to be solved using the SQL language with Oracle SQL Developer. In Part II, based on the MySQL DBMS, the logical schema of a database is provided. You are requested to write the SQL scripts for creating and populating the database and for updating and deleting records.

Part I

Write and execute SQL queries

Write the SQL query in the Worksheet and execute it by clicking on the *Run script* button (see figure).



```
SELECT * FROM DELIVERERS;
```

1 Description of the *Delivery* database

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

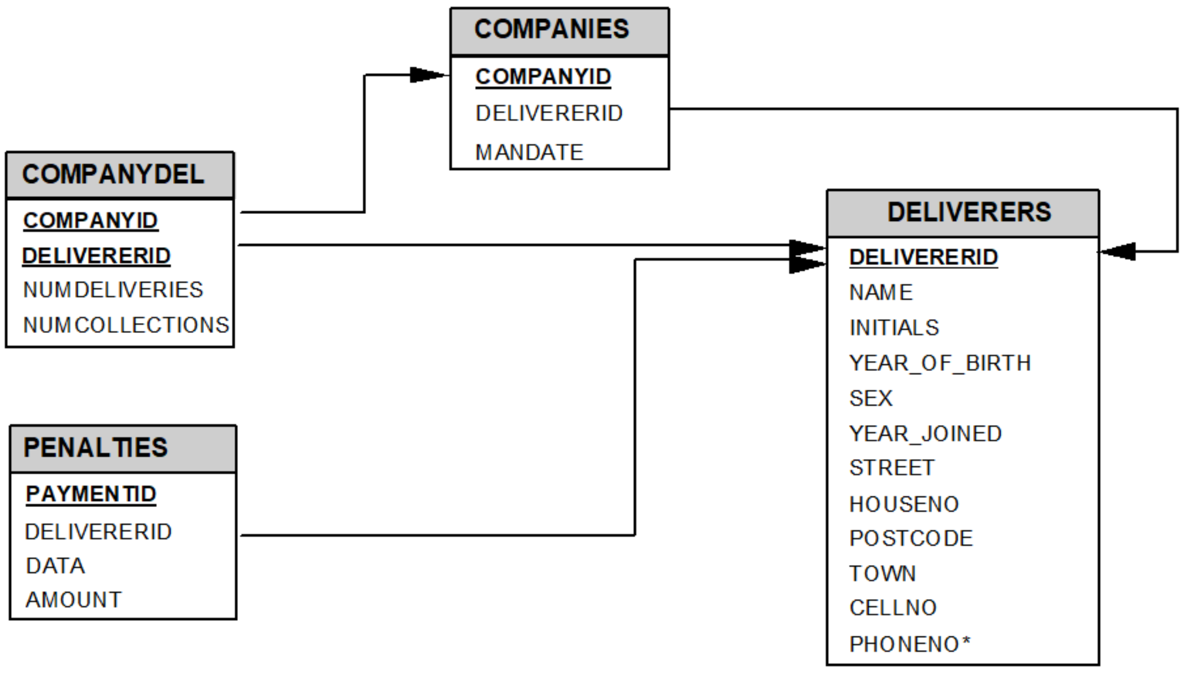
The DELIVERERS table contains the personal data for the deliverers working at the firm. For each deliverer, the following information is available: identification code (DELIVERERID), last name, first name initials, year of birth, sex, year when she/he began working for the firm, street, house number, 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 deliveries (NUMDELIVERIES) and collections (NUMCOLLECTIONS) made by each deliverer for each customer company. Note that the table only reports the deliverer-company 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.

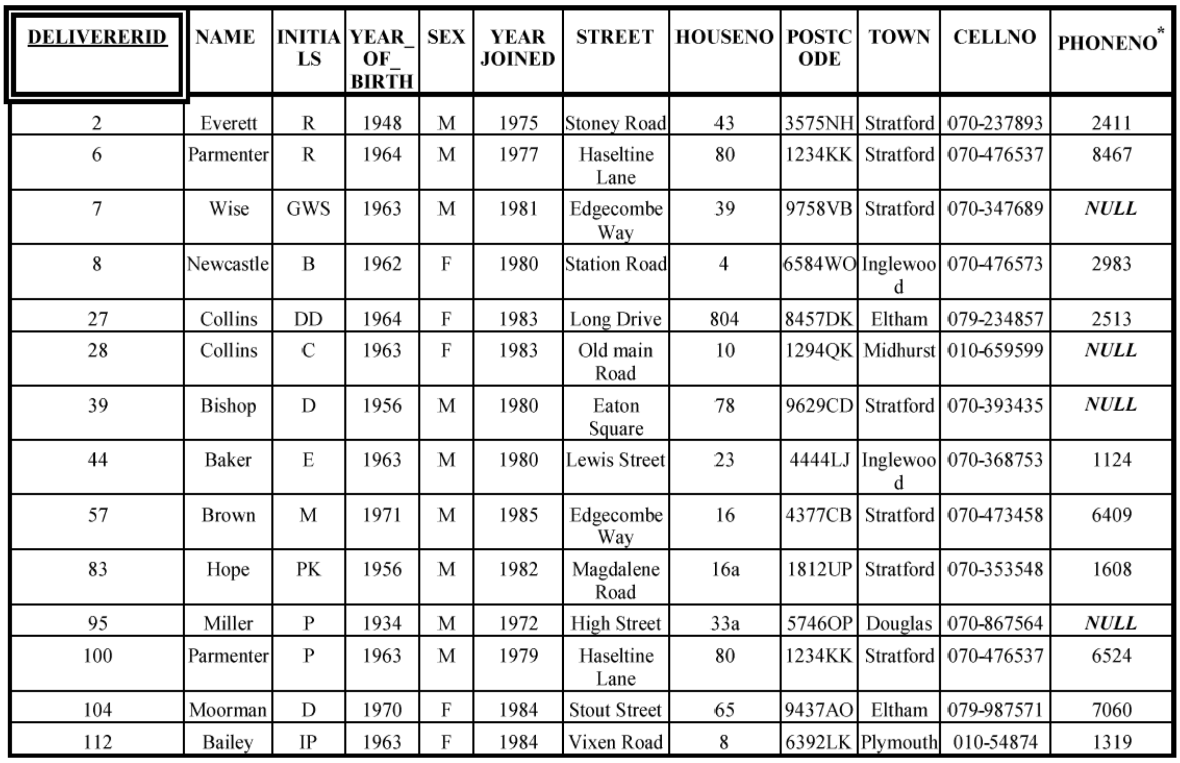
The database schema is shown in the subsequent figure. Section 2 reports the details of every table instance.



2 Table instances for the *Delivery* database

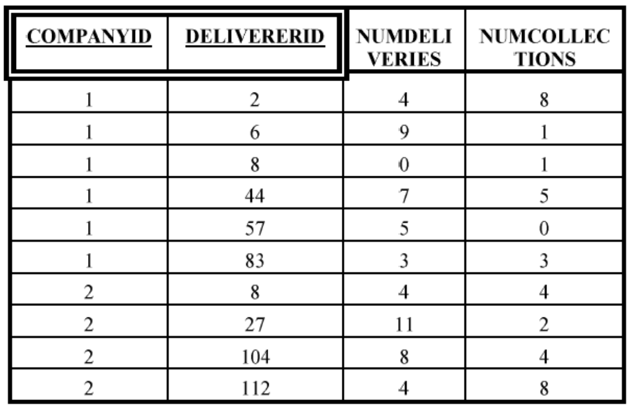
Primary key is underlined. Optional attributes are denoted with *.

DELIVERERS table



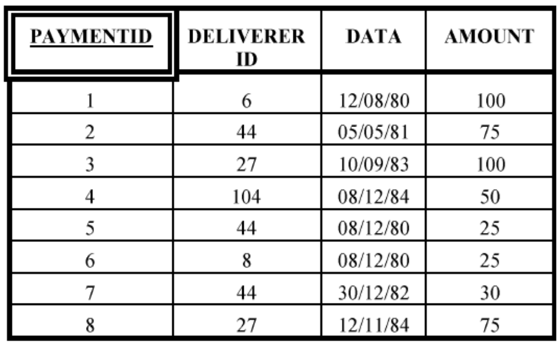
DELIVERERID	NAME	INITIALS	YEAR_OF_BIRTH	SEX	YEAR JOINED	STREET	HOUSENO	POSTCODE	TOWN	CELLNO	PHONENO*
2	Everett	R	1948	M	1975	Stoney Road	43	3575NH	Stratford	070-237893	2411
6	Parmenter	R	1964	M	1977	Haseltine Lane	80	1234KK	Stratford	070-476537	8467
7	Wise	GWS	1963	M	1981	Edgecombe Way	39	9758VB	Stratford	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



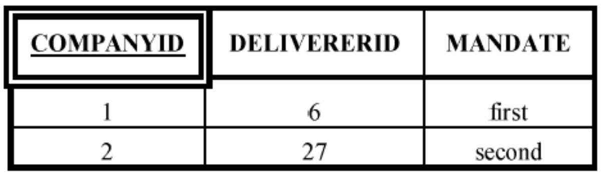
COMPANYID	DELIVERERID	NUMDELIVERIES	NUMCOLLECTIONS
1	2	4	8
1	6	9	1
1	8	0	1
1	44	7	5
1	57	5	0
1	83	3	3
2	8	4	4
2	27	11	2
2	104	8	4
2	112	4	8

PENALTIES table



PAYMENTID	DELIVERER ID	DATA	AMOUNT
1	6	12/08/80	100
2	44	05/05/81	75
3	27	10/09/83	100
4	104	08/12/84	50
5	44	08/12/80	25
6	8	08/12/80	25
7	44	30/12/82	30
8	27	12/11/84	75

***COMPANIES** table*



COMPANYID	DELIVERERID	MANDATE
1	6	first
2	27	second

3 SQL Queries

1. Find the identification code of the deliverer who has received the highest number of fines.
2. Find the identification codes of the deliverers who have *only* delivered (or collected) parcels to (from) firms in which deliverer no. 57 has delivered or collected parcels.
3. Find the identification codes of the deliverers who have delivered (or collected) parcels to (from) *all* of the firms in which deliverer no. 57 has delivered or collected parcels, and *only* to (from) such firms (i.e., to/from no other firms than those visited by deliverer no. 57).
4. 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.
5. 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.
6. 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

4 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.
This part is based on MySQL, in particular the version available in XAMPP.

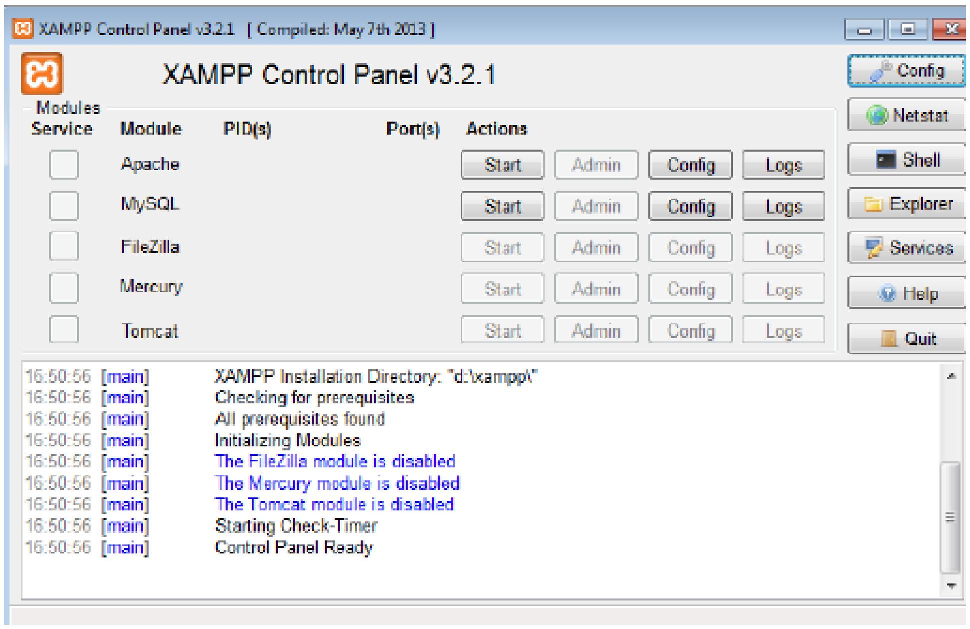
Boot MySQL server on localhost and start Apache

The execution of scripts with SQL commands for the creation and population of the database will be performed through the Web interface of MySQL. Before opening the Web interface of MySQL it is necessary to:

- Start the local Apache server;
- Start the local MySQL server.

Specifically, execute the following steps:

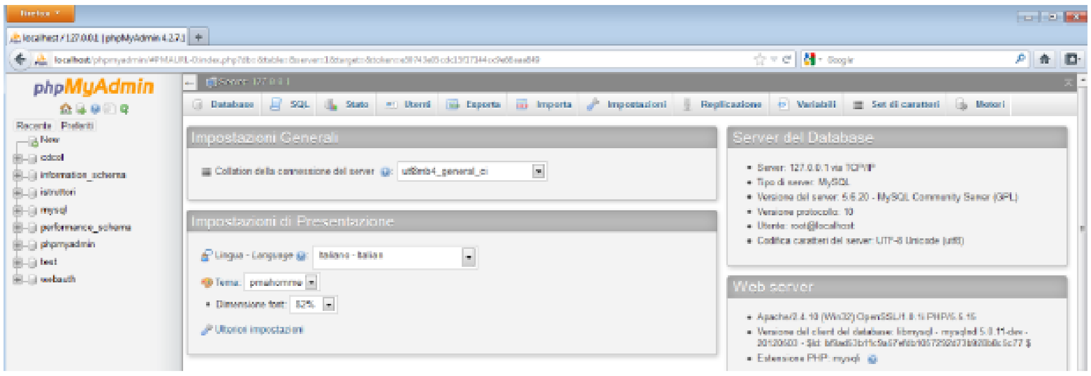
1. Start "XAMPP Control Panel".



```
16:50:56  [main]   XAMPP Installation Directory: "d:\xampp\"
16:50:56  [main]   Checking for prerequisites
16:50:56  [main]   All prerequisites found
16:50:56  [main]   Initializing Modules
16:50:56  [main]   The FileZilla module is disabled
16:50:56  [main]   The Mercury module is disabled
16:50:56  [main]   The Tomcat module is disabled
16:50:56  [main]   Starting Check-Timer
16:50:56  [main]   Control Panel Ready
```

2. Start Apache clicking the Start button in the row of "Apache" module.
3. Start MySQL clicking the Start button in the row of "MySQL" module.

4. Open the MySQL Web interface clicking the Admin button in the row of "MySQL" module (the browser will automatically open the URL associated to the page of administration and SQL querying, i.e., *phpMyAdmin*).



Collation della connessione del server: utf8mb4_general_ci

Lingua - Language: Italiano - Italian

Tema: pmahomme

Dimensione font: 82%

- Server: 127.0.0.1 via TCP/IP
- Tipo di server: MySQL
- Versione del server: 5.6.20 - MySQL Community Server (GPL)
- Versione protocollo: 10
- Utente: root@localhost
- Codifica caratteri del server: UTF-8 Unicode (utf8)

- Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15
- Versione del client del database: libmysql - mysqlnd 5.0.11-dev - 20120503 - $Id: [illegible] $
- Estensione PHP: mysqli

5. To execute a SQL script from the Web interface of MySQL:
   - Select the "Import" panel.
   - Select the file with the script you want to execute and click on "Go" button.
6. To execute the creation/population script more than once, you need to cancel any existing instance of the database, either directly from the "Database" panel or by including at the beginning of the script the commands for deleting the existing tables.

Generation of scripts for the creation and population of DB

1. Scripts are simple text files written with any text editor (e.g., Notepad, Word, Wordpad).
2. Scripts are usually saved with .sql extension.
3. Scripts contain a sequence of instructions ending with semicolon ";".
4. To create a DB in MySQL the following preliminary instructions are needed (at the beginning of the script):
   - SET storage_engine=InnoDB; (*to activate the InnoDB engine for the management of databases*).
   - CREATE DATABASE IF NOT EXISTS DatabaseName; (*creation of the DB named 'DatabaseName' if it doesn't exist*).
   - USE DatabaseName; (*set DatabaseName as current DB*).
5. To activate the automatic check of referential integrity the following command is available:
   - SET FOREIGN_KEY_CHECKS=1; (*enabled*) or 0; (*disabled*).
6. Preliminary instructions are followed by the sequence of SQL commands for the creation and population of DB (CREATE TABLE and INSERT=).
7. Check that syntax and data types are compliant with those required by MySQL.

8. Unless otherwise stated, MySQL always automatically commits executed instructions. To manage transactions the following commands are available:
    - SET autocommit=0; (*disabled*) or 1; (*enabled*)
    - START TRANSACTION;
    - COMMIT; (*to make effective all the instructions of the transaction*)

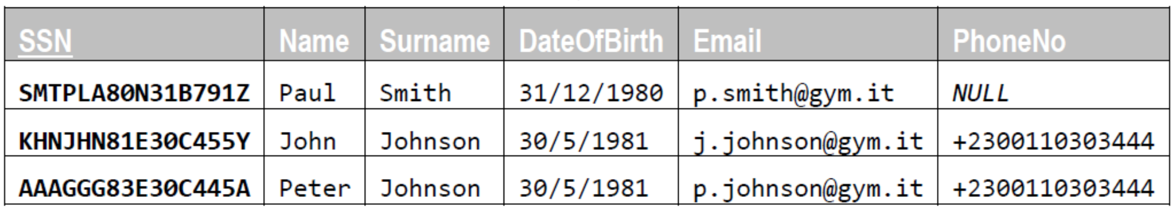
5 Description of the *Gym* database

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 attributed are denoted with *):

- TRAINER(SSN, Name, Surname, DateOfBirth, Email, PhoneNo*)
- COURSE(CId, Name, Type, Level)
- SCHEDULE (*SSN*, Day, StartTime, Duration, *CId*, GymRoom)

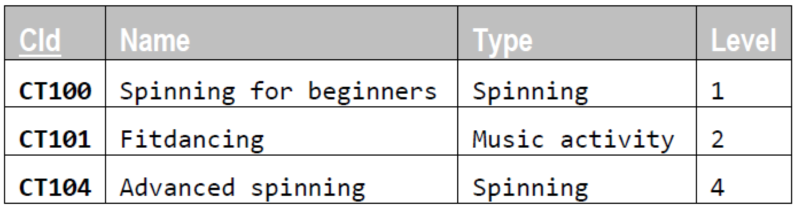
For each trainer, the Social Security Number (SSN), the name, the surname, the date of birth, the email address, and the phone number (if any) are known. For each course, the code, the name, the type, and the level (1-4) are known. The course schedule lists the day of the week and the start time for 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



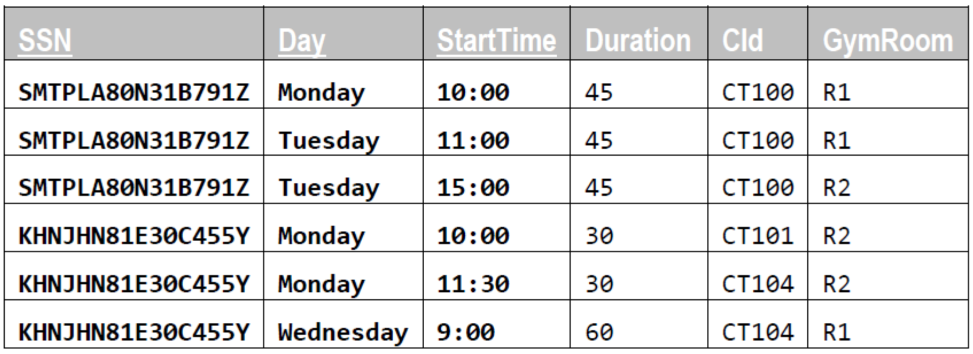
SSN	Name	Surname	DateOfBirth	Email	PhoneNo
SMTPLA80N31B791Z	Paul	Smith	31/12/1980	p.smith@gym.it	*NULL*
KHNJHN81E30C455Y	John	Johnson	30/5/1981	j.johnson@gym.it	+2300110303444
AAAGGG83E30C445A	Peter	Johnson	30/5/1981	p.johnson@gym.it	+2300110303444

COURSE table



CId	Name	Type	Level
CT100	Spinning for beginners	Spinning	1
CT101	Fitdancing	Music activity	2
CT104	Advanced spinning	Spinning	4

SCHEDULE table



SSN	Day	StartTime	Duration	CId	GymRoom
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.

6 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 5. In particular:

   - 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.
   - 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:

   - DROP TABLE SCHEDULE;
   - DROP TABLE COURSE;
   - DROP TABLE TRAINER;

4. Write and execute the following update instructions in SQL, one by one, and check what happens in the database.

   (a) Update the phone number of trainer identified by SSN 'KHNJHN81E30C455Y', setting its value to '+390112333551'.

   (b) Update the database in order to move in room 'R4' all the lessons scheduled in room 'R2'.

   (c) 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 someway related with the policy of constraints management selected during the creation of tables?

   (d) Delete the trainer with SSN 'SMTPLA80N31B791Z'. How are the 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?

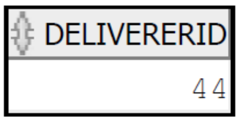
7 Solutions Part I (SQL queries)

1. Find the identification code of the deliverer who has received the highest number of fines.

```
    SELECT DELIVERERID
    FROM PENALTIES
    GROUP BY DELIVERERID
    HAVING COUNT(*) = (SELECT MAX(NumPenalties)
                       FROM (SELECT DELIVERERID, COUNT(*) AS NumPenalties
                             FROM PENALTIES
                             GROUP BY DELIVERERID) TOTMULTEDELIVERERS);
```

   or

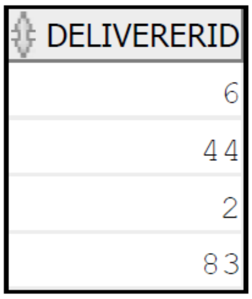
```
    SELECT DELIVERERID
    FROM (SELECT DELIVERERID, COUNT(*) AS NumPenalties
          FROM PENALTIES
          GROUP BY DELIVERERID) TOTMULTEDELIVERERS1
    WHERE NumPenalties = (SELECT MAX(NumPenalties)
                          FROM (SELECT DELIVERERID, COUNT(*) AS NumPenalties
                                FROM PENALTIES
                                GROUP BY DELIVERERID) TOTMULTEDELIVERERS2);
```



DELIVERERID
44

2. Find the identification codes of the deliverers who have *only* delivered (or collected) parcels to (from) firms in which deliverer no. 57 has delivered or collected parcels.

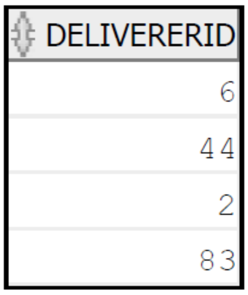
```
    SELECT DISTINCT DELIVERERID
    FROM COMPANYDEL
    WHERE DELIVERERID <> 57
    AND DELIVERERID NOT IN (SELECT DELIVERERID
                            FROM COMPANYDEL
                            WHERE COMPANYID NOT IN
                                (SELECT COMPANYID
                                 FROM COMPANYDEL
                                 WHERE DELIVERERID = 57));
```



DELIVERERID
6
44
2
83

3. Find the identification codes of the deliverers who have delivered (or collected) parcels to (from) *all* of the firms in which deliverer no. 57 has delivered or collected parcels, and *only* to (from) such firms (i.e., to/from no other firms than those visited by deliverer no. 57).

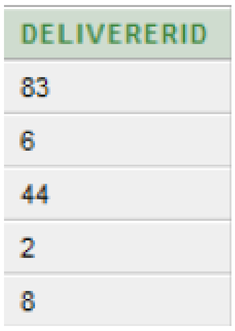
```
SELECT DELIVERERID
FROM COMPANYDEL
WHERE DELIVERERID <> 57
AND DELIVERERID NOT IN (SELECT DELIVERERID
                        FROM COMPANYDEL
                        WHERE COMPANYID NOT IN
                            (SELECT COMPANYID
                             FROM COMPANYDEL
                             WHERE DELIVERERID = 57))
GROUP BY DELIVERERID
HAVING COUNT(*) = (SELECT COUNT(*)
                  FROM COMPANYDEL
                  WHERE DELIVERERID = 57);
```



DELIVERERID
6
44
2
83

4. 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.

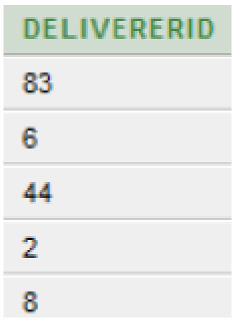
```
SELECT DELIVERERID, MIN(DATA), MAX(DATA)
FROM PENALTIES
GROUP BY DELIVERERID
HAVING COUNT(*) >= 2;
```



DELIVERERID
83
6
44
2
8

5. 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.

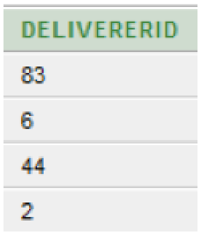
```
SELECT P1.DELIVERERID, DATA, AMOUNT
FROM PENALTIES P1
WHERE P1.DATA = (SELECT MAX(DATA)
                 FROM PENALTIES P2
                 WHERE P2.DELIVERERID = P1.DELIVERERID);
```



DELIVERERID
83
6
44
2
8

6. 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);
```



DELIVERERID
83
6
44
2

8 Solutions 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 5.

```
 -- 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 SCHEDULE;
DROP TABLE IF EXISTS COURSE;
DROP TABLE IF EXISTS TRAINER;

-- create tables
CREATE TABLE TRAINER (
SSN CHAR(20) ,
Name CHAR(50) NOT NULL ,
Surname CHAR(50) NOT NULL ,
DateOfBirth DATE NOT NULL ,
Email CHAR(50) NOT NULL ,
PhoneNo CHAR(20) NULL ,
PRIMARY KEY (SSN),
CONSTRAINT email_constr CHECK (Email LIKE '%@%')
);

CREATE TABLE COURSE (
CId CHAR(10) ,
Name CHAR(50) NOT NULL ,
Type CHAR(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) NOT NULL ,
DayOfWeek CHAR(15) 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.

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

```
UPDATE SCHEDULE
SET GymRoom = 'R4'
WHERE GymRoom = 'R2';
```

```
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.

```
    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.