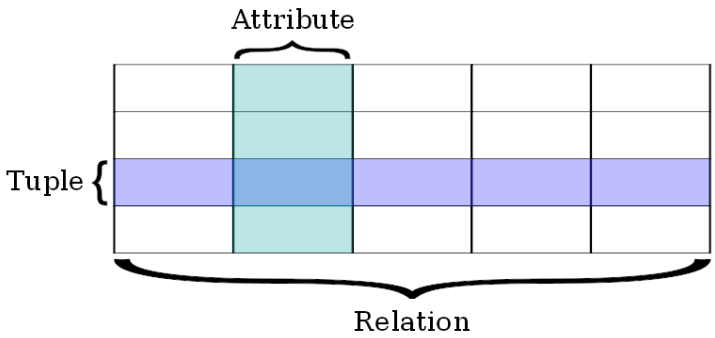
Activities and Materials



Jaume Barceló Vicens DNI 43135949R

Professors d'ensenyament secundari (0590) Especialitat informàtica

Família professional informàtica

Cicle formatiu de grau superior d'administració de sistemes informàtics en xarxa

Mòdul professional de gestió de bases de dades

Codi 0372 170h (+90h d'anglès)

http://github.com/jbarcelo/programacio_didactica_gbd

3 de juliol de 2022

Índex

1 Unit 1: Introduction

Course: Database management
Unit: Introduction to databases
Assignment: Introduction to databases
Teamwork: Groups of two

General instructions for the assignments:

- A single submission for each group.
- Include the name of all the authors in the front page.
- Submit a PDF file.
- Include an index.
- Don't limit yourself to succinctly answer the questions of the assignment. Consider the instructions as a starting point. These instructions are general guidelines to develop your work. The maximum mark for simply following the instructions is a "pass" (5). To obtain better grades, be creative and go beyond the instructions given by the teacher.

Prepare an introductory paper on databases. You should include information about:

- Different alternatives to store data: plain files, spreadsheets, databases...
- What are databases.
- Why are they needed.
- A brief historical approach mentioning the hierarchical model and network model. Also the relational model.
- Differences about centralized databases and distributed databases.
- Examples of databases.
- Examples of database management systems.
- Classify some current database technologies according to their properties (relational/nosql, centralized/distributed, in-memory/on-disk, open-source/proprietary)

Don't limit yourself to the aforementioned aspects. The goal is to write a paper as complete as possible.

Mention the different sources of information you have used to prepare your paper.

It is highly recommended that you read the first chapter of Databases Systems Concepts by Silbershatz et al.

Course: Database management
Unit: Introduction to databases
Assignment: The Role of a DBA
Teamwork: Groups of two

General instructions for the assignments:

- A single submission for each group.
- Include the name of all the authors in the front page.
- Submit a PDF file.
- Include an index.
- Don’t limit yourself to succinctly answer the questions of the assignment. Consider the instructions as a starting point. These instructions are general guidelines to develop your work. The maximum mark for simply following the instructions is a “pass” (5). To obtain better grades, be creative and go beyond the instructions given by the teacher.

Describe the role of the Database Administrator. These are some of the ideas and guidelines for your work.

- Why is it needed?
- Why is it important?
- What responsibilities are involved?
- What kind of organization hires a DBA? Find a job off offer and job description?
- What is the salary range for a DBA?
- Are there specializations of this role?
- Describe some of the tasks performed by a DBA.

Introduction to Databases (TEST)

The respondent's email (**null**) was recorded on submission of this form.

1. Email *

2. What is a database-management system (DBMS)? 1 point

Mark only one oval.

- A piece of software that sits between the operating system and the file system.
- A collection of relations (tables) containing rows and columns.
- A collection of interrelated data and a set of programs to access those data.
- A model to represent abstract data in a computer system.

3. Which of the following is not a DBMS? 1 point

Mark only one oval.

- Oracle
- Docker
- MariaDB
- Postgresql

4. Which of the following was not a problem when data was stored directly into files before the popularization of modern DBMS? 1 point

Mark only one oval.

- The Entity-Relationship Model.
- Concurrent-access anomalies.
- Data isolation.
- Security

5. Which of the following is not one of the data abstraction levels? 1 point

Mark only one oval.

- View level.
- Physical level.
- Logical level.
- Independent level.

6. Which of the following is not one of the models we have seen? 1 point

Mark only one oval.

- Integrative Data Model.
- Object-based Data Model.
- Relational Model.
- Entity-Relationship Model.

7. What does SQL stand for? 1 point

Mark only one oval.

- Structured Query Language
- Storage Query Language
- Spontaneous Query Language
- Simple Query Language

8. What language do we use to specify the database schema? 1 point

Mark only one oval.

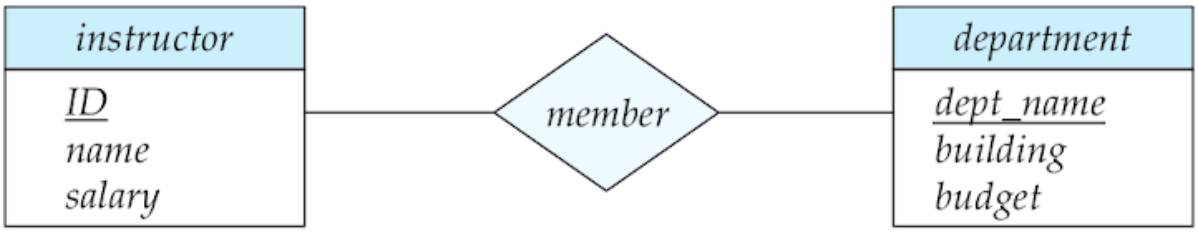
- Database Specification Language
- Database Schema Language
- Data Definition Language
- Data Description Language

9. The characteristic data structure of relational databases is 1 point

Mark only one oval.

- The tree.
- The mesh.
- The list.
- The table.

10. The following picture shows 1 point



Mark only one oval.

- An Entity-Relationship diagram.
- A physical model.
- An object diagram.
- A flow diagram.

11. Atomicity, Consistency, Isolation, Durability (ACID) 1 point

Mark only one oval.

- are the properties of the Relational Model.
- are the properties of physical schema.
- are the properties of database transactions.
- are the properties of object oriented databases.

12. When do we use DDL? 1 point

Mark only one oval.

- When we query data.
- When we insert data.
- When we delete data.
- When we create tables.

13. The action of extracting knowledge from large amounts of stored data is called 1 point

Mark only one oval.

- Data mining
- Networking
- Machine learning
- Data refining

14. Which of the following does not belong to a three-tier architecture database application? 1 point

Mark only one oval.

- Front end user
- Front end client
- Back end application server
- Back end database

2 Unit 2: Introduction to the Relational Model

Course: Database management
Unit: Introduction to the relational model
Assignment: Install a DBMS
Teamwork: Groups of two

Install a DBMS of your choice in a virtual machine. Connect to the database. Document the process and include screenshots.

For example, you can use virtualbox to create a virtual ubuntu machine and install MariaDB. If you want, you can use a different virtualization software, a different OS and a different DBMS.

Course: Database management
Unit: Introduction to the relational model
Assignment: Keys
Teamwork: Groups of two

Consider the relation “student” with three attributes (“id”, “DNI”, “name”). Both the “id” and the “DNI” values are unique for each student. They are never repeated.

1) List all superkeys.
2) List all candidate keys.

Given the following relation R and the set of functional dependencies F that hold on R, find all candidate keys for R.
R (A, B, C, D, E, F)
F = {AB → C, AC → B, AD → E, BC → A, E → F}

Course: Database management
Unit: Introduction to the relational model
Assignment: Relational diagram
Teamwork: Groups of two

Use any tool you like to draw a relational diagram. For example,

https://dbdiagram.io

Draw a relational diagram with, at least, four tables. Include foreign keys.

Document the whole process.

Course: Database management
Unit: Introduction to the relational model
Assignment: Relational algebra
Teamwork: Groups of two
Compute the resulting relations.

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

1.-

- $\sigma_{A=B \wedge D > 5}(r)$

2.-

- $\Pi_{A,C}(r)$

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

3.-

- r ∪ s:

4.-

- $r - s$:

5.-

- $r \cap s$

- Relations r, s:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

6.-

- r x s:

- Relations r

A	B
α	1
β	2

r

7.-

- $r \times \rho_s (r)$

- Relations *r*, *s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

8.-

- $\sigma_{A=C} (r \times s)$

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

9.-

- $r \bowtie s$

10.-

$$\Pi_{A, r.B, C, r.D, E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$$

Course: Database management
Unit: Introduction to the relational model
Test

PART I (50 points):

Draw a relational diagram with at least three tables.

PART II:

Compute the following relational algebra operations.

r:

A	B	C
1	2	3
4	5	6
7	8	9

s:

A	B	C
1	2	1
4	6	5
7	8	9

1.- (4 points) $r \cup s$
2.- (4 points) $r - s$
3.- (4 points) $r \cap s$
4.- (4 points) $\Pi_{B,C}$ (s)
5.- (4 points) $\sigma_{A \neq C}$ (s)
6.- (10 points) $(\Pi_{B,C}$ (s))x($\rho_t(\sigma_{A \neq C}$ (s)))

u:

A	D	E
1	2	1
4	6	5
9	8	7

7.- (10 points) $r \bowtie u$
8.- (10 points) $\Pi_{r.A,B,C,D,E}(\sigma_{r.A=u.A}(r \times u))$

3 Unit 3: Introduction to SQL 1

Course: Database management
Unit: Introduction to SQL
Assignment: Create a Database
Teamwork: Groups of two or three

Use google's cloud shell to create a mysql database with the tables you defined in your relational diagram.
https://console.cloud.google.com/cloudshell?pli=1

Remember that this platform is volatile. It is not persistent. It will not be possible to continue at the point you left it.

Your submission should include a pdf report. In the report, include the relational diagram you did in the previous assignment to make it easier for the reader to understand the database that is created.

Save the scripts to generate the database in a separate file and submit it together with your report. Submit two different files, not a single compressed file.

Course: Database management
Unit: Introduction to SQL
Assignment: Dockerized database
Teamwork: Groups of two or three

If you want to submit more than one file, submit separate files. Please don't compress all of them in a single file.

PART I: Docker

A) Use google's cloud shell (https://console.cloud.google.com/cloudshell?pli=1) to launch a "hello-world" container.
B) Show the list of images in the local repository.
C) Show the list of running containers.
D) Show the list of all containers, including those that are not running.
E) Run a dockerized nginx.
F) In a different console, find out the address of the nginx server.
G) Use to curl to connect to the server and verify that it is working.
H) Look at the list of running containers.
I) Stop the nginx container.
J) Delete all containers.
K) Delete all images.

PART II: Docker database

L) Run a mysql docker database in detached mode. Use an environment variable to set the password.
M) Find the IP address of the database docker.
N) Connect to the database.
O) Create a database.
P) Use the database.
Q) Create a table 'department' with columns 'id', 'name','building' and 'budget'.
R) Insert a row into the table with values 'Physics', 'Watson', '110000.00'.
S) Exit the mysql client.
T) Create a directory named sql.
U) Use mysqldump to dump your database into a file called school.sql in the directory created in the previous step.
V) Create a new container with a volume connecting the directory you created with /docker-entrypoint-initdb.d . This will automatically populate the database of your new container. Run it in detached mode.
W) Find the IP address of the new container.
X) Use the mysql client to connect to the new container.
Y) Verify that the table and its contents are available.
Z) Exit the mysql client.
AA) List all running containers.
BB) Stop all running containers.

CC) Delete the containers.
DD) Delete the directory you created and its contents.

Provide feedback about the difficulties you encountered and the things you have learned.

Definició i execució d’un projecte de Docker

Jaume Barceló Vicens

Definició del projecte

Grup i assignatura

Es tracta del mòdul de primer Gestió de Bases de Dades del cicle de grau superior ASIX. Hi ha dos grups, un d'ASIX normal i l'altre d'ASIX dual. El centre és el CIFP Francesc de Borja Moll.

Context

- Antecedents i coneixements previs de l'alumne.

Els alumnes no tenen coneixements previs. Fa un mes que han començat el cicle i per tant parteixen de zero. Saben algunes comandes bàsiques de SQL i tenen algunes nocions bàsiques de virtualització amb virtualbox.

- Equipament necessari

Els alumnes disposen d'ordinadors portàtils amb un navegador i connexió a internet. S'utilitzarà google cloud shell. S'utilitzarà l'aula virtual google classroom per entregar l'enunciat als alumnes i recollir les seves entregues.

- Descripció del projecte

La descripció del projecte és en anglès ja que es tracta d'un mòdul que s'imparteix en anglès.

Introduction

The purpose of this assignment is to expose first year students to the docker technology and its application to the quick deployment and population of databases.

We also use Google's cloud shell. The advantage is that the students don't need to prepare and run their own virtual machine. The disadvantage is that the environment is volatile and the students have to start from scratch every time they connect to this service.

The assignment is divided into two parts that are independent. The first part is intended to be a first contact with docker while the second is where the student learns about the use of environment variables and volumes.

Goals

After completing this assignment the students will be able to

- Download an image and run a container.
- Use environment variables to configure a dockerized database.
- Create a docker volume to populate the database when it is started.

Task

Work in groups of two or three people.

If you want to submit more than one file, submit separate files. Please don't compress all of them in a single file.

PART I: Docker

A. Use google's cloud shell (https://console.cloud.google.com/cloudshell?pli=1) to launch a "hello-world" container.
B. Show the list of images in the local repository.
C. Show the list of running containers.
D. Show the list of all containers, including those that are not running.
E. Run a dockerized nginx.
F. In a different console, find out the address of the nginx server.
G. Use to curl to connect to the server and verify that it is working.
H. Look at the list of running containers.
I. Stop the nginx container.
J. Delete all containers.
K. Delete all images.

PART II: Docker database

L. Run a mysql docker database in detached mode. Use an environment variable to set the password.
M. Find the IP address of the database docker.
N. Connect to the database.
O. Create a database.
P. Use the database.
Q. Create a table 'department' with columns 'id', 'name','building' and 'budget'.
R. Insert a row into the table with values 'Physics', 'Watson', '110000.00'.

S. Exit the mysql client.
T. Create a directory named sql.
U. Use mysqldump to dump your database into a file called school.sql in the directory created in the previous step
V. Create a new container with a volume connecting the directory you created with /docker-entrypoint-initdb.d . This will automatically populate the database of your new container. Run it in detached mode.
W. Find the IP address of the new container.
X. Use the mysql client to connect to the new container.
Y. Verify that the table and its contents are available.
Z. Exit the mysql client.
AA. List all running containers.
BB. Stop all running containers.
CC. Delete the containers.
DD. Delete the directory you created and its contents.

Provide feedback about the difficulties you encountered and the things you have learned.

Evaluation

Criteria	Bad	Acceptable	Good
Technical competence	The student cannot complete the task on its own and requires continuous help from the teacher.	The student can complete the task with occasional help of the classmates.	The student can complete the task without requiring any help.
Report	The student does not produce a report that allows the teacher to verify that the task has been completed.	The student produces a report that allows the teacher to verify that the task has been completed.	The student produces a report that would allow someone else to easily follow the steps and understand what is done.
Insight	The student reproduces steps and does not understand their meaning.	The explanations in the report show that the student understands the concepts.	The report provides explanations that will help others to understand the concepts.

Cross-evaluation: After they have completed the task, the students will have the chance to comment on the strengths and weaknesses of the assignment to allow future improvements.

- Solució base

A. Command: `jbarcelo@cloudshell:~$ docker hello-world`
B. Command: `jbarcelo@cloudshell:~$ docker image ls`
C. Command: `jbarcelo@cloudshell:~$ docker ps`
D. Command: `jbarcelo@cloudshell:~$ docker ps -a`
E. Command: `jbarcelo@cloudshell:~$ docker run nginx`
F. Command: `jbarcelo@cloudshell:~$ docker inspect 98b|grep '"IPAddress"'`
G. Command: `jbarcelo@cloudshell:~$ curl 172.18.0.3`
H. Command: `jbarcelo@cloudshell:~$ docker ps`
I. Command: `jbarcelo@cloudshell:~$ docker stop 98b`
J. Command: `jbarcelo@cloudshell:~$ docker rm 44a 98b` ,
K. Command: `jbarcelo@cloudshell:~$ docker rmi hello-world nginx`
L. Command: `jbarcelo@cloudshell:~$ docker run --name test-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:latest`
M. Command: `jbarcelo@cloudshell:~$ docker inspect 53d | grep '"IPAddress"'`
N. Command: `jbarcelo@cloudshell:~$ mysql -u root -p -h 172.18.0.2`
O. Command: `mysql> CREATE DATABASE school;`
P. Command: `mysql> USE school;`
Q. Command: `mysql> CREATE TABLE department (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(20), building VARCHAR(20), budget DECIMAL(10,2));`
R. Command: `mysql> INSERT INTO department (name, building, budget) VALUES ('Physics', 'Whatson', 110000.00);`
S. Command: `mysql> exit`
T. Command: `jbarcelo@cloudshell:~$ mkdir sql`
U. Command: `jbarcelo@cloudshell:~$ mysqldump school -u root -p -h 172.18.0.2 > sql/school.sql`
V. Command: `jbarcelo@cloudshell:~$ docker run -v /home/jbarcelo/sql:/docker-entrypoint-initdb.d -e MYSQL_ROOT_PASSWORD=my-secret-pw -e MYSQL_DATABASE=school -d mysql`
W. Command: `jbarcelo@cloudshell:~$ docker inspect 36e | grep '"IPAddress"'`
X. Command: `jbarcelo@cloudshell:~$ mysql -u root -p -h 172.18.0.3`
Y. Command: `mysql> select * from school.department;`
Z. Command: `mysql> exit`
AA. Command: `jbarcelo@cloudshell:~$ docker ps`
BB. Command: `jbarcelo@cloudshell:~$ docker stop 36e 53d`
CC. Command: `jbarcelo@cloudshell:~$ docker rm 36e 53d`
DD. Command: `jbarcelo@cloudshell:~$ rm -rf sql`

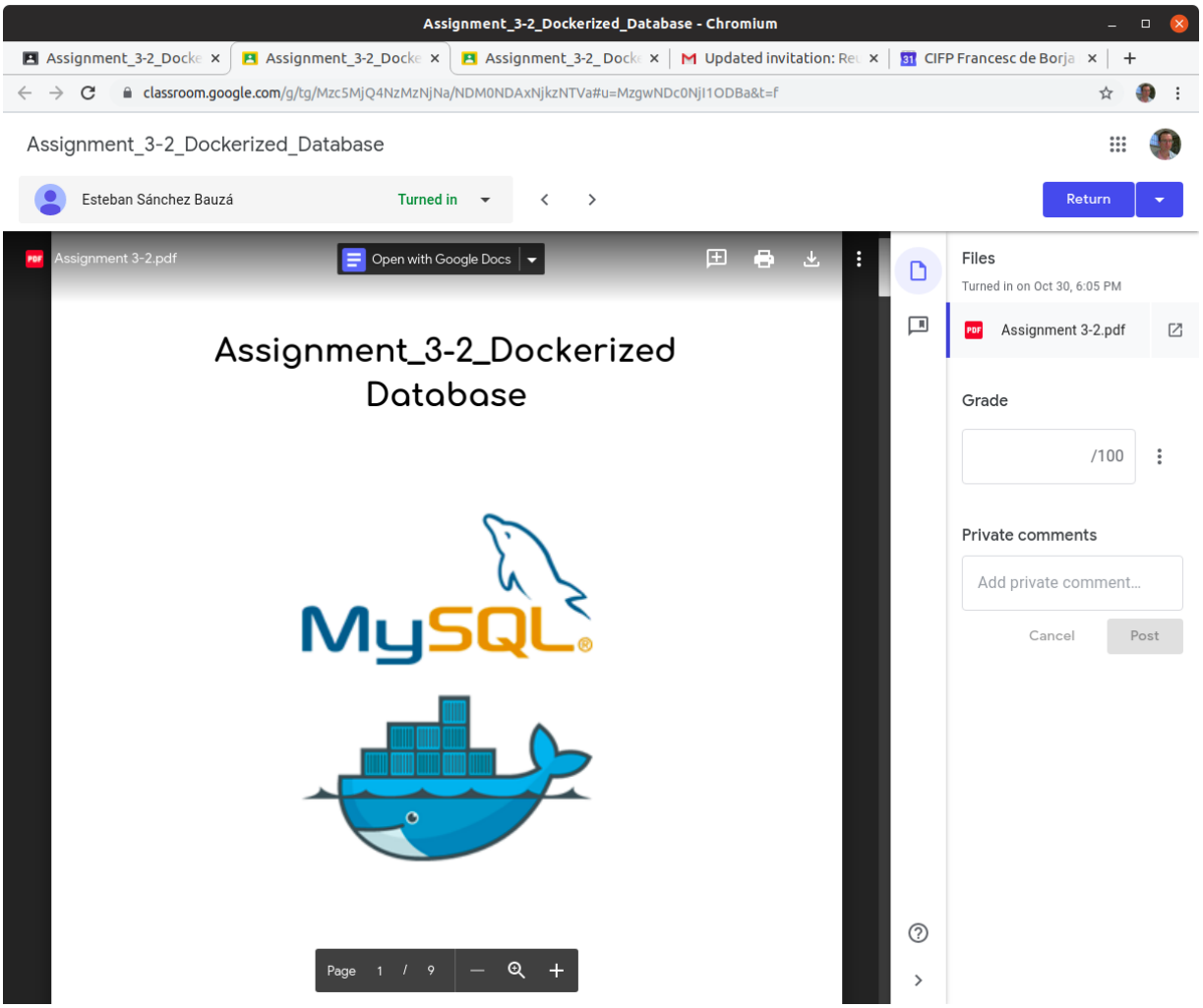
- Conclusions

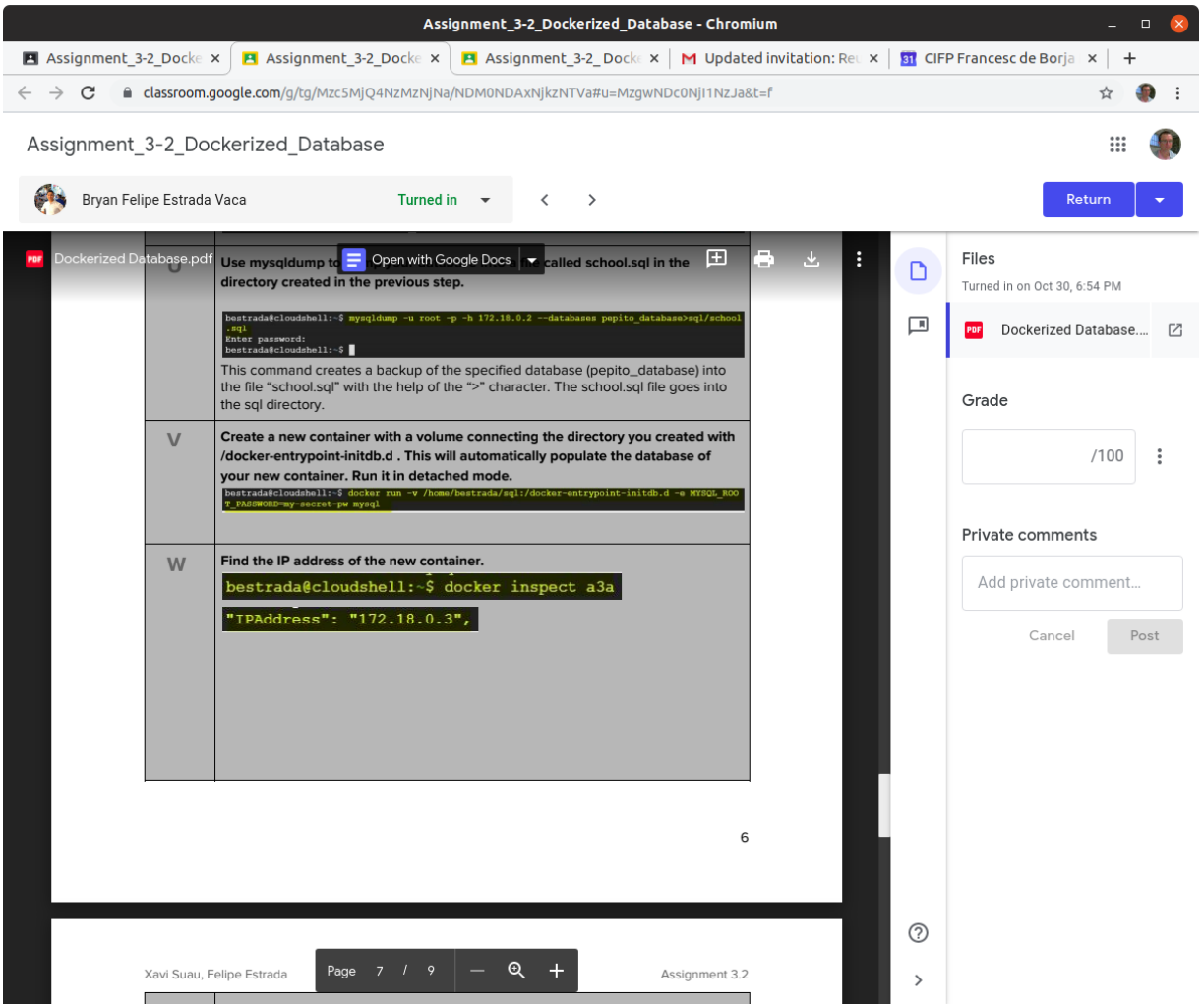
El projecte proposat està pensat com un primer contacte amb docker en el context de bases de dades. Donades les circumstàncies, pens que és el més adequat. Els alumnes aprendran les comandes relacionades amb el cicle de vida dels contenidors. A més, també s'utilitzaran variables d'entorn i volums. Més endavant i en altres assignatures, ja tendran l'oportunitat d'aprofundir en els seus coneixements de docker.

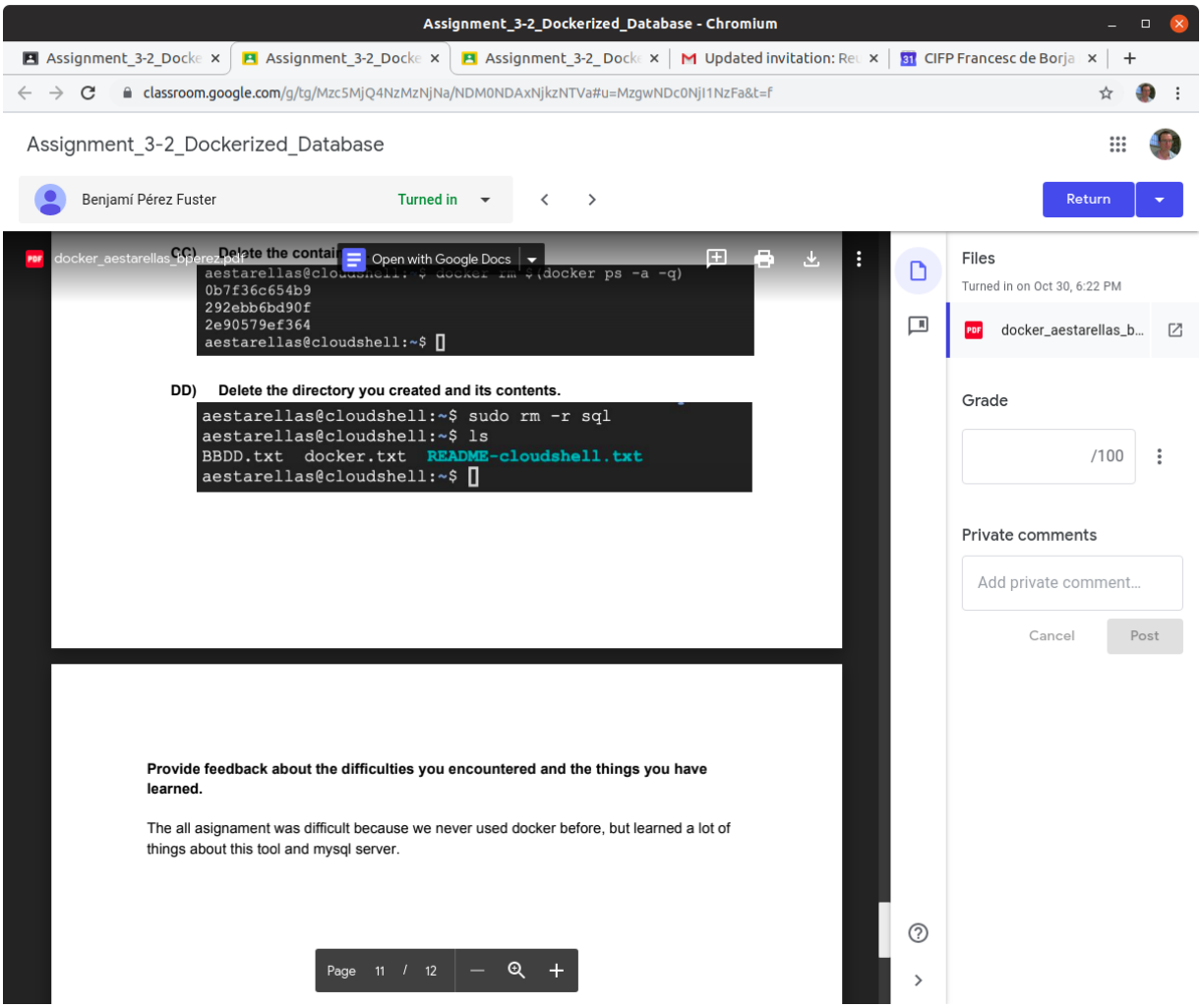
Execució del projecte

El projecte s'ha executat la setmana del 28 d'octubre de 2019. El fet més important a destacar és que la majoria dels estudiants no han pogut completar la tasca en el temps previst. Tot i això, el resultat ha estat satisfactori ja que han après els principis bàsics de docker. Es tractava d'un exercici introductori i es pot dir que ha aconseguit els seus objectius. La part que ha necessitat explicació ha estat la de la creació d'un volum per tal que el contenidor tengués accés a una carpeta de l'amfitrió.

A continuació s'adjunten algunes captures de pantalla de les entregues realitzades pels alumnes.


Assignment_3-2_Dockerized_Database - Chromium
classroom.google.com/g/tg/Mzc5MjQ4NzMzNjNa/NDM0NDAxNjkzNTVa#u=MzgwNDc0NjI1ODBa&t=f
Assignment_3-2_Dockerized_Database
Esteban Sánchez Bauzá
Turned in
Return
Assignment 3-2.pdf
Open with Google Docs
Assignment_3-2_Dockerized Database
MySQL
Page 1 / 9
Files
Turned in on Oct 30, 6:05 PM
Assignment 3-2.pdf
Grade
/100
Private comments
Add private comment...
Cancel
Post


Assignment_3-2_Dockerized_Database - Chromium
classroom.google.com/g/tg/Mzc5MjQ4NzMzNjNa/NDM0NDAxNjkzNTVa#u=MzgwNDc0NjI1NzJa&t=f
Assignment_3-2_Dockerized_Database
Bryan Felipe Estrada Vaca
Turned in
Return
Dockerized Database.pdf
Open with Google Docs
Use mysqldump to
called school.sql in the directory created in the previous step.
bestrada@cloudshell:~$ mysqldump -u root -p -h 172.18.0.2 --databases pepito_database>sql/school.sql
Enter password:
bestrada@cloudshell:~$
This command creates a backup of the specified database (pepito_database) into the file "school.sql" with the help of the ">" character. The school.sql file goes into the sql directory.
V
Create a new container with a volume connecting the directory you created with /docker-entrypoint-initdb.d . This will automatically populate the database of your new container. Run it in detached mode.
bestrada@cloudshell:~$ docker run -v /home/bestrada/sql:/docker-entrypoint-initdb.d -e MYSQL_ROOT_PASSWORD=my-secret-pw mysql
W
Find the IP address of the new container.
bestrada@cloudshell:~$ docker inspect a3a
"IPAddress": "172.18.0.3",
6
Xavi Suau, Felipe Estrada
Page 7 / 9
Assignment 3.2
Files
Turned in on Oct 30, 6:54 PM
Dockerized Database....
Grade
/100
Private comments
Add private comment...
Cancel
Post


Assignment_3-2_Dockerized_Database - Chromium
classroom.google.com/g/tg/Mzc5MjQ4NzMzNjNa/NDM0NDAxNjkzNTVa#u=MzgwNDc0NjI1NzFa&t=f
Assignment_3-2_Dockerized_Database
Benjamí Pérez Fuster
Turned in
Return
docker_aestarellas_bperez.pdf
Open with Google Docs
aestarellas@cloudshell:~$ docker rm $(docker ps -a -q)
0b7f36c654b9
292ebb6bd90f
2e90579ef364
aestarellas@cloudshell:~$
DD) Delete the directory you created and its contents.
aestarellas@cloudshell:~$ sudo rm -r sql
aestarellas@cloudshell:~$ ls
BBDD.txt docker.txt README-cloudshell.txt
aestarellas@cloudshell:~$
Provide feedback about the difficulties you encountered and the things you have learned.
The all asignament was difficult because we never used docker before, but learned a lot of things about this tool and mysql server.
Page 11 / 12
Files
Turned in on Oct 30, 6:22 PM
docker_aestarellas_b...
Grade
/100
Private comments
Add private comment...
Cancel
Post

Course: Database management
Unit: Introduction to SQL
Assignment: School Database Script
Teamwork: Groups of two or three

Write an SQL script for MySQL (or MariaDB) that creates a database for a university with two related tables: instructor and department. Use the data in the slides of the first lesson for a description of the tables. Fill the tables with the data in the slides of the first lesson.

There are two alternative ways to accomplish this goal.

A) Create the script directly and then verify that it is working executing the script in a DBMS.
B) Create the database, the tables and the data and then use mysqldump to obtain the script.

You can use a virtual machine, google cloudshell, docker or any other alternative you like for your MariaDB or MySQL DBMS.

Deliver the resulting script and a report as two separate files. Don’t use a compressing or archiving software to combine the two files in a single one.

Feel free to ask any for any clarifications or further instructions if needed.

Course: Database management
Unit: Introduction to SQL
Material: Requirements for the Test

1 A virtual machine with ubuntu, ubuntu server or similar.
2 Docker
3 The docker image for mysql
4 The sql script to generate the database https://pastebin.com/raw/MR4JL3ZR

CIFP Francesc de Borja Moll - ASIX
Course: Database management
Unit: Introduction to SQL
Test

Create a PDF document with screenshots proving that you have followed each of the following 20 steps. Use numbers in the PDF document to show which screenshot correspond to each step.

1.- Create a docker container with an image of mysql. Use an environment variable to set the password and run the container in detached mode.
2.- Find the ip address of the container.
3.- Connect to the mysql server.
4.- Create a database. Create the tables department and instructor. Including primary keys and foreign keys. department(dept_name, building, budget), instructor(ID, name, dept_name, salary).
5.- Fill in the tables with data. At least one row for each table.
6.- Delete a row from a table.
7.- Add a column to a table.
8.- Drop a column from a table.
9.- Exit the mysql client.
10.- Stop the container.
11.- Delete the container.
12.- Create a mysql docker container with a volume to map a folder with the university.sql script to /docker-entrypoint-initdb.d
13.- Verify that you have the university database and all the tables.
14.- Show the name of the instructors of the Comp. Sci. department with a salary lower than 80000.
15.- List the name of the instructors and the building they are working in.
16.- List the names of the departments that don't have the highest budget. Avoid the repetition of names in the list.
17.- Create the emp_super table. Select the supervisee of the supervisee of the supervisee of Mary.

person	supervisor
Bob	Alice
Mary	Susan
Alice	David

David	Mary

18.- List the names of the instructors that finish with the letter ‘n’.
19.- List the names of the instructors that have four or more letters.
20.- List the names of the departments in order, from the highest budget to the lowest.

4 Unit 4: Introduction to SQL 2

Course: Database management
Unit: Introduction to SQL II
Material: Queries MariaDB

Create the database EMPLOYEESDB with the tables DEPARTMENTS and EMPLOYEES from this script.
https://drive.google.com/file/d/1C5XKPuh4CtrJ4Rqy7PN-Vo9zdfXR25dB/view?usp=sharing

```
-- phpMyAdmin SQL Dump
-- version 4.9.0.1
-- https://www.phpmyadmin.net/
--
-- Servidor: localhost
-- Tiempo de generación: 19-09-2019 a las 18:02:35
-- Versión del servidor: 10.3.15-MariaDB-1
-- Versión de PHP: 7.3.4-2

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `EMPLOYEESDB`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `DEPARTMENTS`
--

CREATE TABLE `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Volcado de datos para la tabla `DEPARTMENTS`
--

INSERT INTO `DEPARTMENTS` (`num`, `name`) VALUES
```

```
(10, 'ACCOUNTING'),
(20, 'RESEARCH'),
(30, 'SALES'),
(40, 'PRODUCTION');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `EMPLOYEES`
--

CREATE TABLE `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `occupation` varchar(30) DEFAULT NULL,
  `manager` int(11) DEFAULT NULL,
  `begin_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Volcado de datos para la tabla `EMPLOYEES`
--

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `occupation`,
`manager`, `begin_date`, `salary`, `commission`, `dept_num`) VALUES
(1000, 'PITT', 'BRAD', 'OWNER', NULL, '1984-01-01', 104000, NULL, 20),
(7369, 'REDLEAF', 'JANE', 'EMPLOYEE', 8001, '1990-12-17', 104000, NULL,
20),
(7499, 'DERN', 'BRUCE', 'SALESMAN', 7698, '1990-02-20', 15000, 390, 30),
(7521, 'ROBINSON', 'SARAH', 'SALESMAN', 7782, '1991-02-22', 16250, 650,
30),
(7566, 'DI CAPRIO', 'LEONARDO', 'MANAGER', 1000, '1991-04-02', 29000,
NULL, 20),
(7654, 'HERRIMAN', 'DAMON', 'SALESMAN', 7698, '1991-09-29', 16000, 1020,
30),
(7698, 'BRONSON', 'CHARLES', 'MANAGER', 1000, '1991-05-01', 30050, NULL,
30),
(7782, 'ROBBIE', 'MARGOT', 'MANAGER', 1000, '1991-06-09', 28850, NULL,
10),
(7788, 'MADISON', 'MIKEY', 'ANALYST', 8000, '1991-11-09', 30000, NULL,
20),
(7844, 'DUNHAM', 'LENA', 'SALESMAN', 7698, '1991-09-08', 13500, 0, 30),
```

```
(7876, 'RITTEN', 'REBECCA', 'EMPLOYEE', 7788, '1991-09-23', 14300, NULL,
20),
(7900, 'COLLINS', 'CLIFTON', 'EMPLOYEE', 8001, '1991-12-03', 13350, NULL,
30),
(7902, 'ROWLING', 'KANSAS', 'ANALYST', 8000, '1991-12-03', 30000, NULL,
20),
(7934, 'HARRIS', 'DANIELLE', 'EMPLOYEE', 8001, '1992-01-23', 16900, NULL,
10),
(8000, 'QUALLEY', 'MARGARET', 'MANAGER', 1000, '1991-01-09', 28850, NULL,
20),
(8001, 'FANNING', 'DAKOTA', 'MANAGER', 1000, '1992-06-10', 28850, NULL,
20);

--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `DEPARTMENTS`
--
ALTER TABLE `DEPARTMENTS`
  ADD PRIMARY KEY (`num`);

--
-- Indices de la tabla `EMPLOYEES`
--
ALTER TABLE `EMPLOYEES`
  ADD PRIMARY KEY (`num`),
  ADD KEY `dept_num` (`dept_num`),
  ADD KEY `manager` (`manager`);

--
-- Restricciones para tablas volcadas
--

--
-- Filtros para la tabla `EMPLOYEES`
--
ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES
`DEPARTMENTS` (`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`manager`) REFERENCES
`EMPLOYEES` (`num`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Do the following queries:
1. Show the last name, occupation and department number of each employee.
2. Show the number and name of each department.
3. Show all the data of all the employees.
4. Employee data sorted by last name ascendent.
5. Employee data sorted by department number descending.
6. Employee data sorted by department number descending and, within each department, sort data by employee surname ascending.
7. Show the data of the employees whose salary is greater than 20000.
8. Show the data of the employees whose occupation is 'SALESMAN'.
9. Select the surname and occupation of the employees of department number 20.
10. Select employees whose occupation is 'SALESMAN'. Show data sorted by surname.
11. Show employees whose department is 20 or 30 and whose occupation is 'MANAGER'. Sort the result by dept_num descendent and surname ascendent.
12. Show employees who have a salary greater than 20000 or who belong to the department number 20.
13. Show employees sorted by their occupation and by their surname.
14. Select employees from the EMPLOYEES table whose surname starts with 'S'.
15. Select from the EMPLOYEE table those rows whose SURNAME starts with 'C 'and the occupation has an 'E' in any position.
16. Select employees whose salary is between 10000 and 20000.
17. Obtain the employees whose occupation is 'SALESMAN' and have a commission exceeding 1000.
18. Number and surnames of the employees whose surname ends with 'S' and have a salary higher than 30000.
19. Data of the departments whose name starts with 'A' and ends with ‘G’.
20. Show the surnames of the employees who do not have a commission (clue: is NULL).
21. Show the surnames of the employees who do not have a commission and whose last name begins with 'H' (clue: is NULL).
22. Show the surnames of the employees whose occupation is 'SALESMAN', 'MANAGER' or 'EMPLOYEE'.
23. Show the surnames of the employees whose occupation is neither "MANAGER" nor "EMPLOYEE", and also have a salary higher than 20000.
24. Select the surname, salary and department number of employees whose salary is greater than 20000 in departments 10 or 30.
25. Show the surname and number of employees whose salary is not between 10000 and 20000.
26. Get the surnames of all employees in lowercase.
27. In a query, concatenate the name of each employee with his/her surname.
28. Show the surname and the length of the surname (function LENGTH) of all employees,

sorted by the length of the surnames of the employees descending.
29. Show data of the employees whose surname have 4 characters and ends with ‘N’. Do a version using the function LENGTH and another one without it.
30. Obtain the years of recruitment of all employees (YEAR function) but without duplicated data. Order the results.
31. Show the data of the employees that have been recruited in the year 1992.
32. Show data of employees who have been recruited in the month of February of any year (function MONTHNAME).
33. Show the data of the employees whose surname begins with 'R' and have been recruited in the year 1990.
34. Show the name, surname, department name of the employees who have no commission (clue: is NULL).
35. Show the name, surname and occupation of employees with the name, surname and occupation of their managers. Columns must have different names and you must order it by the manager surname.

Course: Database management
Unit: Introduction to SQL II
Material: Queries MariaDB 2

Check the following database:

DEPARTMENTS:

num	name	town_code
10	ACCOUNTING	SVQ
20	RESEARCH	MAD
30	SALES	BCN
40	PRODUCTION	BIO

EMPLOYEES:

num	surname	name	manager	start_date	salary	commission	dept_num	occu_code
800	BANDERAS	ANTONIO	7839	1991-01-09	2885	NULL	20	MAN
7369	SÁNCHEZ	SERGIO	7902	1990-12-17	1040	NULL	20	EMP
7499	ARROYO	MARTA	7698	1990-02-20	1500	390	30	SAL
7521	SALA	RAUL	7698	1991-02-22	1625	650	30	SAL
7566	JIMÉNEZ	JUDIT	7839	1991-04-02	2900	NULL	20	MAN
7654	MARTÍN	MONICA	7698	1991-09-29	1600	1020	30	SAL
7698	NEGRO	BARTOLOME	7839	1991-05-01	3005	NULL	30	MAN
7782	CEREZO	ENRIQUE	7839	1991-06-09	2885	NULL	10	MAN
7788	GIL	JESUS	7566	1991-11-09	3000	NULL	20	ANA
7844	TOVAR	LUIS	7698	1991-09-08	1350	0	30	SAL
7876	ALONSO	FERNANDO	7788	1991-09-23	1430	NULL	20	EMP
7900	JIMENO	XAVIER	7698	1991-12-03	1335	NULL	30	EMP
7902	FERNÁNDEZ	ANA	7566	1991-12-03	3000	NULL	20	ANA
7934	MUÑOZ	ANTONIA	7782	1992-01-23	1690	NULL	10	EMP
8001	RUIZ	FERNANDA	7839	1992-06-10	2885	NULL	20	MAN

OCCUPATIONS:

code	name
ANA	ANALYST
EMP	EMPLOYEE
MAN	MANAGER
PRE	PRESIDENT
SAL	SALESMAN

TOWNS:

code	name
BCN	BARCELONA
BIO	BILBAO
MAD	MADRID
SVQ	SEVILLA

Import the next database:

```
CREATE DATABASE IF NOT EXISTS `EMPLOYEESDBNORMAL`;
USE `EMPLOYEESDBNORMAL`;

CREATE TABLE IF NOT EXISTS `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
```

```
  `town_code` varchar(3) DEFAULT NULL,
  PRIMARY KEY (`num`),
  KEY `town_code` (`town_code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
INSERT INTO `DEPARTMENTS` (`num`, `name`, `town_code`) VALUES
(10, 'ACCOUNTING', 'SVQ'),
(20, 'RESEARCH', 'MAD'),
(30, 'SALES', 'BCN'),
(40, 'PRODUCTION', 'BIO');

CREATE TABLE IF NOT EXISTS `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `manager` int(11) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL,
  `occu_code` varchar(3) DEFAULT NULL,
  PRIMARY KEY (`num`),
  KEY `dept_num` (`dept_num`),
  KEY `occu_code` (`occu_code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `manager`, `start_date`, `salary`,
`commission`, `dept_num`, `occu_code`) VALUES
(800, 'BANDERAS', 'ANTONIO', 7839, '1991-01-09', 2885, NULL, 20,
'MAN'), (7369, 'SÁNCHEZ', 'SERGIO', 7902, '1990-12-17', 1040, NULL, 20,
'EMP'), (7499, 'ARROYO', 'MARTA', 7698, '1990-02-20', 1500, 390, 30,
'SAL'), (7521, 'SALA', 'RAUL', 7698, '1991-02-22', 1625, 650, 30,
'SAL'), (7566, 'JIMÉNEZ', 'JUDIT', 7839, '1991-04-02', 2900, NULL, 20,
'MAN'), (7654, 'MARTÍN', 'MONICA', 7698, '1991-09-29', 1600, 1020, 30,
'SAL'), (7698, 'NEGRO', 'BARTOLOME', 7839, '1991-05-01', 3005, NULL,
30, 'MAN'), (7782, 'CEREZO', 'ENRIQUE', 7839, '1991-06-09', 2885, NULL,
10, 'MAN'), (7788, 'GIL', 'JESUS', 7566, '1991-11-09', 3000, NULL, 20,
'ANA'), (7844, 'TOVAR', 'LUIS', 7698, '1991-09-08', 1350, 0, 30,
'SAL'),
(7876, 'ALONSO', 'FERNANDO', 7788, '1991-09-23', 1430, NULL, 20,
'EMP'), (7900, 'JIMENO', 'XAVIER', 7698, '1991-12-03', 1335, NULL, 30,
'EMP'), (7902, 'FERNÁNDEZ', 'ANA', 7566, '1991-12-03', 3000, NULL, 20,
'ANA'), (7934, 'MUÑOZ', 'ANTONIA', 7782, '1992-01-23', 1690, NULL, 10,
'EMP'), (8001, 'RUIZ', 'FERNANDA', 7839, '1992-06-10', 2885, NULL, 20,
'MAN');

CREATE TABLE IF NOT EXISTS `OCCUPATIONS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL,
  PRIMARY KEY (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `OCCUPATIONS` (`code`, `name`) VALUES
('ANA', 'ANALYST'),
('EMP', 'EMPLOYEE'),
('MAN', 'MANAGER'),
('PRE', 'PRESIDENT'),
('SAL', 'SALESMAN');

CREATE TABLE IF NOT EXISTS `TOWNS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL,
```

```
  PRIMARY KEY (`code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `TOWNS` (`code`, `name`) VALUES
('BCN', 'BARCELONA'),
('BIO', 'BILBAO'),
('MAD', 'MADRID'),
('SVQ', 'SEVILLA');

ALTER TABLE `DEPARTMENTS`
  ADD CONSTRAINT `DEPARTMENTS_ibfk_1` FOREIGN KEY (`town_code`) REFERENCES `TOWNS`
(`code`);

ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES `DEPARTMENTS`
(`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`occu_code`) REFERENCES `OCCUPATIONS`
(`code`);
```

Do the following queries with that database:

1. Display the number of employees in each department. Use GROUP BY to group by department.

```
+----------+--------------+
| dept_num | N_employees |
+----------+--------------+
|       10 |            2 |
|       20 |            7 |
|       30 |            6 |
+----------+--------------+
3 rows in set (0.001 sec)
```

2. For each occupations obtain the average of salary.

```
+----------+-----------------+
| name     | average_salary |
+----------+-----------------+
| ANALYST  |      3000.0000 |
| EMPLOYEE |      1373.7500 |
| MANAGER  |      2912.0000 |
| SALESMAN |      1518.7500 |
+----------+-----------------+
4 rows in set (0.001 sec)
```

3. Display the departments with more than 5 employees. Use GROUP BY to group by department and HAVING to establish the condition on the groups.

```
+----------+---------------+
| dept_num | num_employees |
+----------+---------------+
|       20 |             7 |
|       30 |             6 |
+----------+---------------+
2 rows in set (0.001 sec)
```

4. Find the average wages (=”*media de los salarios*”) of each department (use the function avg and GROUP BY).

dept_num	average_wages
10	2287.5000
20	2448.5714
30	1735.8333

3 rows in set (0.002 sec)

5. Display the surname of the salesmen of the 'SALES' department.

surname
ARROYO
SALA
MARTÍN
TOVAR

4 rows in set (0.001 sec)

6. Display the sum of salaries of the 'SALES' department.

name	total
SALES	10415

1 row in set (0.001 sec)

7. Display the count of employees with occupation “EMPLOYEE” in every department (show the name of the department).

name	num
ACCOUNTING	1
RESEARCH	2
SALES	1

3 rows in set (0.001 sec)

8. Show the number of different occupations in each department.

Department	Occupation	Number_of_employees
ACCOUNTING	EMPLOYEE	1
ACCOUNTING	MANAGER	1
RESEARCH	ANALYST	2
RESEARCH	EMPLOYEE	2
RESEARCH	MANAGER	3
SALES	EMPLOYEE	1
SALES	MANAGER	1
SALES	SALESMAN	4

8 rows in set (0.004 sec)

9. Show departments that have more than two people working in the same occupation.

Department
RESEARCH
SALES

2 rows in set (0.002 sec)

10. Displays a query that is the union between the table OCCUPATIONS and TOWNS.

code	name
ANA	ANALYST
EMP	EMPLOYEE
MAN	MANAGER
PRE	PRESIDENT
SAL	SALESMAN
BCN	BARCELONA
BIO	BILBAO
MAD	MADRID
SVQ	SEVILLA

9 rows in set (0.001 sec)

11. Do the same query than in exercise 10 but order the results by name.

code	name
ANA	ANALYST
BCN	BARCELONA
BIO	BILBAO
EMP	EMPLOYEE
MAD	MADRID
MAN	MANAGER
PRE	PRESIDENT
SAL	SALESMAN
SVQ	SEVILLA

9 rows in set (0.001 sec)

12. Select the occupation names of all the employees of the department with name 'RESEARCH' and do the union of this query with the selection of the occupation names of the employees of the department with name 'SALES'. Use union operator.

name
ANALYST
EMPLOYEE
MANAGER
SALESMAN

4 rows in set (0.001 sec)

13. Repeat the last query showing the repeated results (union all).

```
+----------+
| name     |
+----------+
| ANALYST  |
| ANALYST  |
| EMPLOYEE |
| EMPLOYEE |
| MANAGER  |
| MANAGER  |
| MANAGER  |
| EMPLOYEE |
| MANAGER  |
| SALESMAN |
| SALESMAN |
| SALESMAN |
| SALESMAN |
+----------+
13 rows in set (0.001 sec)
```

14. Display the number of sellers in the 'SALES' department.

```
+------------------+
| number_of_sellers |
+------------------+
|                4 |
+------------------+
1 row in set (0.001 sec)
```

15. Display the surnames and occupations of the employees of the 'SALES' department.

```
+---------+----------+
| surname | name     |
+---------+----------+
| JIMENO  | EMPLOYEE |
| NEGRO   | MANAGER  |
| ARROYO  | SALESMAN |
| SALA    | SALESMAN |
| MARTÍN  | SALESMAN |
| TOVAR   | SALESMAN |
+---------+----------+
6 rows in set (0.001 sec)
```

16. Display the number of employees and occupations of the employees of the 'SALES' department.

```
+----------+---------------------+
| name     | number_of_employees |
+----------+---------------------+
| EMPLOYEE |                   1 |
| MANAGER  |                   1 |
| SALESMAN |                   4 |
+----------+---------------------+
3 rows in set (0.001 sec)
```

17. Display the number of employees of each department whose profession is "EMPLOYEE".

```
+------------+---------------------+
| name       | number_of_employees |
+------------+---------------------+
| ACCOUNTING |                   1 |
| RESEARCH   |                   2 |
| SALES      |                   1 |
+------------+---------------------+
3 rows in set (0.001 sec)
```

18. Display de department names and the count of employees working into them.

```
+------------+---------------------+
| name       | number_of_employees |
+------------+---------------------+
| ACCOUNTING |                   2 |
| RESEARCH   |                   7 |
| SALES      |                   6 |
+------------+---------------------+
3 rows in set (0.001 sec)
```

19. Display the maximum number of employees of all the departments (clue: you need exercise 18 as a subquery and you should use MAX function).

```
+------------+
| max_number |
+------------+
|          7 |
+------------+
1 row in set (0.001 sec)
```

20. Show the departments whose average salary is greater than the average of salaries of all employees.

```
+----------+----------------+
| dept_num | average_salary |
+----------+----------------+
|       10 |      2287.5000 |
|       20 |      2448.5714 |
+----------+----------------+
2 rows in set (0.001 sec)
```

21. DANGER, this is for PROS: Display the name of the department with more employees and their number of employees (clue you must use HAVING with a subselect inside).

```
+----------+---------------+
| name     | num_employees |
+----------+---------------+
| RESEARCH |             7 |
+----------+---------------+
1 row in set (0.001 sec)
```

22. Repeat 12 changing “union” for “intersect”.

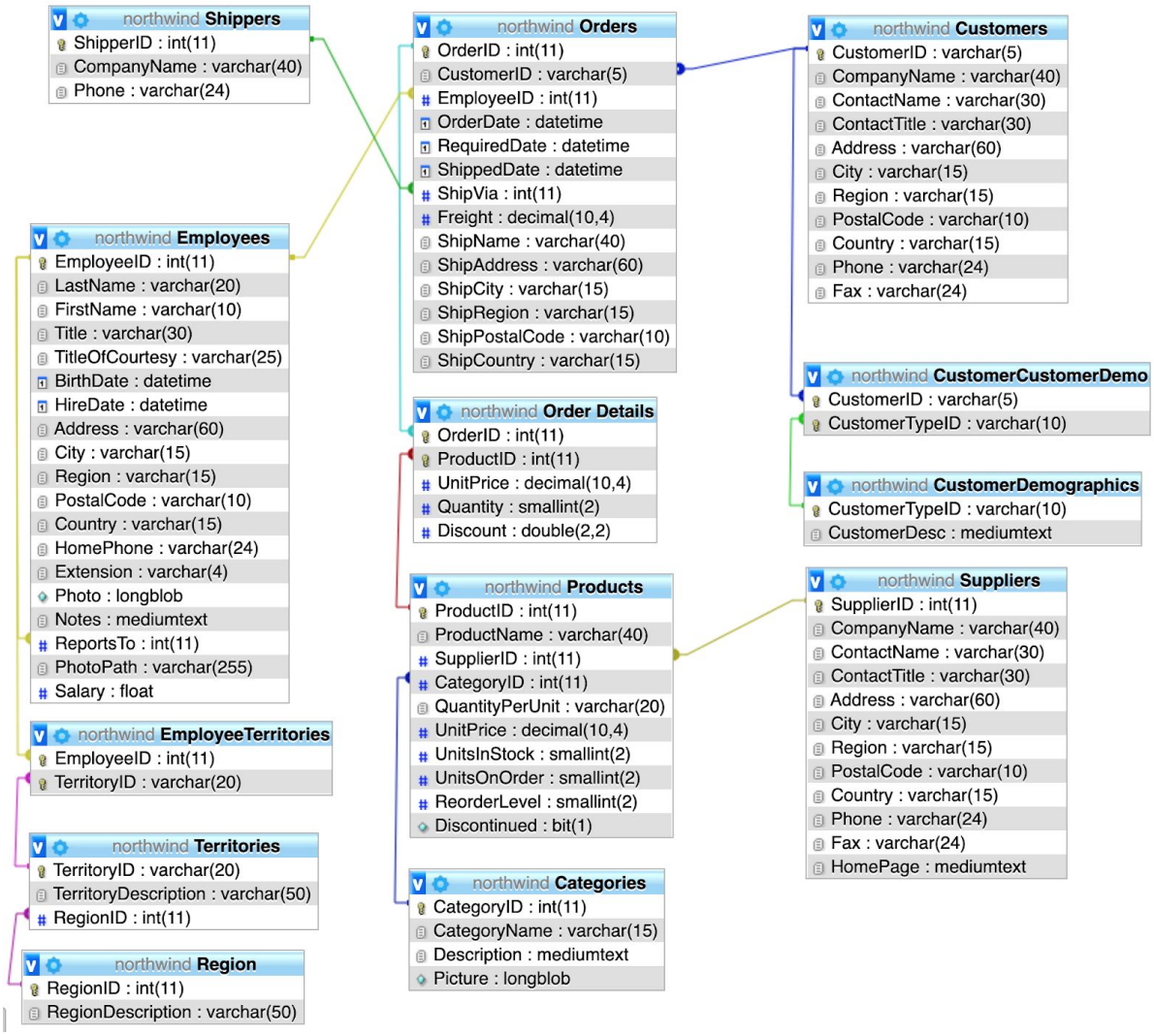
```
+----------+
| name     |
+----------+
| EMPLOYEE |
| MANAGER  |
+----------+
2 rows in set (0.002 sec)
```

23. Repeat 22 but do not use intersect operator to query the same data (clue: IN operator).

```
+----------+
| name     |
+----------+
| EMPLOYEE |
| MANAGER  |
+----------+
2 rows in set (0.002 sec)
```

Course: Database management
Unit: Introduction to SQL II
Material: Northwind

NORTHWIND



Download the script to create the database here:

https://drive.google.com/file/d/1u_H6YXfmoCoGOuqZKv4vnRU_1CCwRLsr/view?usp=sharing

The exercises are not sorted by order of difficulty...

1.- Show total cost of every order in May 1998 (also called “subtotal”).
Clue: price*quantity*(1-discount)

OrderID	OrderDate	Subtotal
11064	1998-05-01 00:00:00	9208.4850
11065	1998-05-01 00:00:00	492.4920
11066	1998-05-01 00:00:00	1811.0625
11067	1998-05-04 00:00:00	169.3575
11068	1998-05-04 00:00:00	4650.3600
11069	1998-05-04 00:00:00	702.0000
11070	1998-05-05 00:00:00	3653.3250
11071	1998-05-05 00:00:00	994.5000
11072	1998-05-05 00:00:00	10175.1000
11073	1998-05-05 00:00:00	585.0000
11074	1998-05-06 00:00:00	476.3850
11075	1998-05-06 00:00:00	1142.7000
11076	1998-05-06 00:00:00	2061.1500
11077	1998-05-06 00:00:00	2417.4930

14 rows in set (0.001 sec)

2.- Show the total sales by year.

OrderYear	Subtotal
1996	438818.2500
1997	1283858.0625
1998	915791.1360

3 rows in set (0.010 sec)

3.- Total sales by country.

ShipCountry	Subtotal
Argentina	15832.2450
Austria	272018.4285
Belgium	68513.2110
Brazil	224188.5360
Canada	107901.4950
Denmark	67825.3875
Finland	38567.9775
France	166722.5820
Germany	477049.2285
Ireland	111768.9105
Italy	32575.0425
Mexico	46943.2275
Norway	11183.5425
Poland	6887.3025
Portugal	24313.8675
Spain	37892.1855
Sweden	116071.2150
Switzerland	64193.0250
UK	118202.1945
USA	513955.6110
Venezuela	118589.0355

21 rows in set (0.008 sec)

4.- Show employee sales order by sales amount

descendent.

LastName	FirstName	EmployeeID	Subtotal
Peacock	Margaret	4	487859.4435
Leverling	Janet	3	415097.5140
Davolio	Nancy	1	393351.2700
Fuller	Andrew	2	346611.0570
King	Robert	7	275527.1805
Callahan	Laura	8	258721.0860
Dodsworth	Anne	9	161779.8000
Suyama	Michael	6	152248.7850
Buchanan	Steven	5	147271.3125

9 rows in set (0.007 sec)

5.- Show employee sales by country order by country ascendent and sales amount descendent.

LastName	FirstName	EmployeeID	ShipCountry	Subtotal
Callahan	Laura	8	Argentina	5363.4750
King	Robert	7	Argentina	2994.8100
Peacock	Margaret	4	Argentina	2592.3300
Dodsworth	Anne	9	Argentina	1841.7750
Davolio	Nancy	1	Argentina	1339.0650

...

Buchanan	Steven	5	USA	31494.6450
Leverling	Janet	3	Venezuela	25349.8635
Callahan	Laura	8	Venezuela	20135.4270
Davolio	Nancy	1	Venezuela	19467.0450
Peacock	Margaret	4	Venezuela	16421.7300
King	Robert	7	Venezuela	13992.6150
Buchanan	Steven	5	Venezuela	10049.5200
Suyama	Michael	6	Venezuela	6598.4100
Fuller	Andrew	2	Venezuela	5837.3250
Dodsworth	Anne	9	Venezuela	737.1000

167 rows in set (0.012 sec)

6.- Sales per category.

CategoryID	CategoryName	Subtotal
1	Beverages	558499.2075
2	Condiments	221511.7125
3	Confections	344943.0855
4	Dairy Products	489632.4420
5	Grains/Cereals	196372.8585
6	Meat/Poultry	346658.2275
7	Produce	205168.3822
8	Seafood	275681.5327

8 rows in set (0.012 sec)

7.- Number of products by category.

CategoryID	CategoryName	Number
1	Beverages	12
2	Condiments	12
3	Confections	13
4	Dairy Products	10
5	Grains/Cereals	7
6	Meat/Poultry	6
7	Produce	5
8	Seafood	12

8 rows in set (0.001 sec)

8.- Order details of the order number 10248.

OrderID	ProductID	ProductName	UnitPrice	Quantity	Discount	ExtendedPrice
10248	11	Queso Cabrales	27.3000	12	0.10	294.84
10248	42	Singaporean Hokkien Fried Mee	19.1100	10	0.10	171.99
10248	72	Mozzarella di Giovanni	67.8600	5	0.10	305.37

3 rows in set (0.002 sec)

9.- Total amount for order 10248.

TotalPrice
772.2000

1 row in set (0.003 sec)

10.- Order heading for order 10248.

OrderID	OrderDate	RequiredDate	CustomerID	CompanyName	Address	City	Region	PostalCode	Country	ShippedDate	CompanyName	Freight	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry	SalesPerson
10248	1996-07-04 00:00:00	1996-08-01 00:00:00	VINET	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	NULL	51100	France	1996-07-16 00:00:00	Federal Shipping	32.3800	59 rue de l-Abbaye	Reims	NULL	51100	France	Steven Buchanan

1 row in set (0.001 sec)

11.- Top 10 products with more units sold.

ProductName	Units
Camembert Pierrot	1577
Raclette Courdavault	1496
Gorgonzola Telino	1397
Gnocchi di nonna Alice	1263
Pavlova	1158
Rhnbru Klosterbier	1155
Guaran Fantstica	1125
Boston Crab Meat	1103
Tarte au sucre	1083
Chang	1057

10 rows in set (0.003 sec)

12.- List of products with its price and its prices in orders. In the table Products UnitPrice is the cost price of the products (what we paid to the providers), in Order Details we have the price that we sold the products to the customers.

ProductID	UnitPrice	UnitPriceSold
1	18.0000	28.0800
1	18.0000	35.1000
2	19.0000	29.6400
2	19.0000	37.0500
3	10.0000	15.6000
3	10.0000	19.5000
4	22.0000	34.3200
4	22.0000	42.9000
5	21.3500	33.1500
5	21.3500	41.6325
6	25.0000	39.0000
6	25.0000	48.7500
...		
70	15.0000	29.2500
71	21.5000	33.5400
71	21.5000	41.9250
72	34.8000	67.8600
72	34.8000	54.2100
73	15.0000	23.4000
73	15.0000	29.2500
74	10.0000	15.6000
74	10.0000	19.5000
75	7.7500	12.0900
75	7.7500	15.1125
76	18.0000	28.0800
76	18.0000	35.1000
77	13.0000	20.2800
77	13.0000	25.3500

156 rows in set (0.008 sec)

13.- Ten most expensive products.

ProductID	ProductName	UnitPrice
38	Cte de Blaye	263.5000
29	Thringer Rostbratwurst	123.7900
9	Mishi Kobe Niku	97.0000
20	Sir Rodney's Marmalade	81.0000
18	Carnarvon Tigers	62.5000
59	Raclette Courdavault	55.0000
51	Manjimup Dried Apples	53.0000
62	Tarte au sucre	49.3000
43	Ipoh Coffee	46.0000
28	Rssle Sauerkraut	45.6000

10 rows in set (0.000 sec)

14.- Top ten products that gave more profit per unit (with year in which they were sold and ordered by profit).

ProductID	year	UnitPrice	UnitPrice	profit
38	1997	263.5000	513.8250	250.3250
38	1998	263.5000	513.8250	250.3250
38	1996	263.5000	411.0600	147.5600
38	1997	263.5000	411.0600	147.5600
29	1998	123.7900	241.3905	117.6005
29	1997	123.7900	241.3905	117.6005
9	1997	97.0000	189.1500	92.1500
9	1998	97.0000	189.1500	92.1500
20	1998	81.0000	157.9500	76.9500
20	1997	81.0000	157.9500	76.9500

10 rows in set (0.002 sec)

15. Top ten products in profit in sales.

ProductID	ProductName	Total_Profit
38	Cte de Blaye	128308.6900
29	Thringer Rostbratwurst	78014.7025
59	Raclette Courdavault	66497.2000
60	Camembert Pierrot	44081.6800
62	Tarte au sucre	43772.5050
51	Manjimup Dried Apples	40265.2660
56	Gnocchi di nonna Alice	39992.3400
17	Alice Mutton	31048.2900
18	Carnarvon Tigers	28322.5000
28	Rssle Sauerkraut	23203.9200

10 rows in set (0.005 sec)

16.- Products above average price (ordered by price descendent).

ProductID	ProductName	UnitPrice
38	Cte de Blaye	263.5000
29	Thringer Rostbratwurst	123.7900
9	Mishi Kobe Niku	97.0000
20	Sir Rodney's Marmalade	81.0000
18	Carnarvon Tigers	62.5000
59	Raclette Courdavault	55.0000
51	Manjimup Dried Apples	53.0000
62	Tarte au sucre	49.3000
43	Ipoh Coffee	46.0000
28	Rssle Sauerkraut	45.6000
27	Schoggi Schokolade	43.9000
63	Vegie-spread	43.9000
8	Northwoods Cranberry Sauce	40.0000
17	Alice Mutton	39.0000
12	Queso Manchego La Pastora	38.0000
56	Gnocchi di nonna Alice	38.0000
69	Gudbrandsdalsost	36.0000
72	Mozzarella di Giovanni	34.8000
60	Camembert Pierrot	34.0000
64	Wimmers gute Semmelkndel	33.2500
53	Perth Pasties	32.8000
32	Mascarpone Fabioli	32.0000
26	Gumbr Gummibrchen	31.2300
10	Ikura	31.0000
7	Uncle Bob's Organic Dried Pears	30.0000

25 rows in set (0.001 sec)

17.- Describe what the following query does.

```
select c.CategoryName as "Product Category",
    case when s.Country in
           ('UK','Spain','Sweden','Germany','Norway',
            'Denmark','Netherlands','Finland','Italy','France')
        then 'Europe'
        when s.Country in ('USA','Canada','Brazil')
        then 'America'
        else 'Asia-Pacific'
     end as "Supplier Continent",
     sum(p.UnitsInStock) as UnitsInStock
from Suppliers s, Products p, Categories c
```

```
where p.SupplierID=s.SupplierID and
        c.CategoryID=p.CategoryID
group by c.CategoryName,
      case when s.Country in
           ('UK','Spain','Sweden','Germany','Norway',
            'Denmark','Netherlands','Finland','Italy','France')
         then 'Europe'
         when s.Country in ('USA','Canada','Brazil')
         then 'America'
         else 'Asia-Pacific'
      end;
```

18.- Top 10 suppliers with more products in our system ordered by number of products descendent and company name ascendent.

```
+------------+--------------------------------+-----+
| SupplierID | CompanyName                    | qty |
+------------+--------------------------------+-----+
|          7 | Pavlova, Ltd.                  |   5 |
|         12 | Plutzer Lebensmittelgromrkte AG |   5 |
|          2 | New Orleans Cajun Delights     |   4 |
|          8 | Specialty Biscuits, Ltd.       |   4 |
|         16 | Bigfoot Breweries              |   3 |
|          1 | Exotic Liquids                 |   3 |
|         14 | Formaggi Fortini s.r.l.        |   3 |
|         24 | G'day, Mate                    |   3 |
|          3 | Grandma Kelly's Homestead      |   3 |
|         11 | Heli Swaren GmbH & Co. KG      |   3 |
+------------+--------------------------------+-----+
10 rows in set (0.001 sec)
```

19.- Number of customers from Barcelona.

```
+------------+
| NumberBCN  |
+------------+
|          1 |
+------------+
1 row in set (0.001 sec)
```

20.- Make a complete list of customers with more than 25 orders placed.

```
+--------------------+-----------+
| CompanyName        | NumOrders |
+--------------------+-----------+
| Ernst Handel       |        30 |
| QUICK-Stop         |        28 |
| Save-a-lot Markets |        31 |
+--------------------+-----------+
3 rows in set (0.002 sec)
```

21.- Customer who has placed the maximum number of orders.

```
+--------------------+-----------+
| CompanyName        | NumOrders |
+--------------------+-----------+
| Save-a-lot Markets |        31 |
+--------------------+-----------+
1 row in set (0.002 sec)
```

22.- Make a list of shippers and the number of orders handled by each.

```
+------------------+-----------+
| CompanyName      | NumOrders |
+------------------+-----------+
| United Package   |       326 |
| Federal Shipping |       255 |
| Speedy Express   |       249 |
+------------------+-----------+
3 rows in set (0.002 sec)
```

23. Make a list of the employees (full name) and for each employee show the number of times that employee has used each of the shippers.

```
+------------------+------------------+----------+
| FullName         | CompanyName      | NumTimes |
+------------------+------------------+----------+
| Buchanan, Steven | Federal Shipping |       13 |
| Buchanan, Steven | Speedy Express   |       14 |
| Buchanan, Steven | United Package   |       15 |
| Callahan, Laura  | Federal Shipping |       29 |
| Callahan, Laura  | Speedy Express   |       27 |
| Callahan, Laura  | United Package   |       48 |
| Davolio, Nancy   | Federal Shipping |       41 |
| Davolio, Nancy   | Speedy Express   |       38 |
| Davolio, Nancy   | United Package   |       44 |
| Dodsworth, Anne  | Federal Shipping |       14 |
| Dodsworth, Anne  | Speedy Express   |       10 |
| Dodsworth, Anne  | United Package   |       19 |
| Fuller, Andrew   | Federal Shipping |       25 |
| Fuller, Andrew   | Speedy Express   |       35 |
| Fuller, Andrew   | United Package   |       36 |
| King, Robert     | Federal Shipping |       28 |
| King, Robert     | Speedy Express   |       20 |
| King, Robert     | United Package   |       24 |
| Leverling, Janet | Federal Shipping |       46 |
| Leverling, Janet | Speedy Express   |       36 |
| Leverling, Janet | United Package   |       45 |
| Peacock, Margaret | Federal Shipping |       40 |
| Peacock, Margaret | Speedy Express   |       46 |
| Peacock, Margaret | United Package   |       70 |
| Suyama, Michael  | Federal Shipping |       19 |
| Suyama, Michael  | Speedy Express   |       23 |
| Suyama, Michael  | United Package   |       25 |
+------------------+------------------+----------+
27 rows in set (0.007 sec)
```

24. Make a list of products that are out of stock.

```
+-----------+------------------------+
| ProductID | ProductName            |
+-----------+------------------------+
|        17 | Alice Mutton           |
|         5 | Chef Anton's Gumbo Mix |
|        31 | Gorgonzola Telino      |
|        53 | Perth Pasties          |
|        29 | Thringer Rostbratwurst |
+-----------+------------------------+
5 rows in set (0.001 sec)
```

25. Make a list of products that are out of stock and have not been discontinued.

Include the suppliers' names.

```
+-----------+------------------+-------------------------+
| ProductID | ProductName      | CompanyName             |
+-----------+------------------+-------------------------+
|        31 | Gorgonzola Telino | Formaggi Fortini s.r.l. |
+-----------+------------------+-------------------------+
1 row in set (0.007 sec)
```

26.- Make a list of products that need to be re-ordered i.e. where the units in stock (the number you have) and the units on order (the number you have coming) is less than the reorder level (the number that prompts you to order more).

```
+-----------+----------------------+
| ProductID | ProductName          |
+-----------+----------------------+
|        30 | Nord-Ost Matjeshering |
|        70 | Outback Lager        |
+-----------+----------------------+
2 rows in set (0.003 sec)
```

27. Top 10 orders with bigger discount (as an amount).

```
+--------------------+-----------+
| CompanyName        | NumOrders |
+--------------------+-----------+
| Save-a-lot Markets |        31 |
+--------------------+-----------+
1 row in set (0.002 sec)
```

CIFP Francesc de Borja Moll - ASIX
Course: Database management
Unit: Introduction to SQL 2
Test

1.- [university] Find all departments where the total salary is greater than the average of the total salary at all departments.

dept_name
Comp. Sci.
Finance
Physics

2.- [university] Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building.

3.- [university] Make each student in the Music department who has earned more than 144 credit hours an instructor in the Music department, with a salary of $18,000

4.- [university] All instructors with salary over $100,000 receive a 3 percent raise, whereas all others receive a 5 percent raise. Use a single update statement.

5.- [employees] Show the number employees of different occupations in each department.

Department	Occupation	Number_of_employees
ACCOUNTING	EMPLOYEE	1
ACCOUNTING	MANAGER	1
RESEARCH	ANALYST	2
RESEARCH	EMPLOYEE	2
RESEARCH	MANAGER	3
SALES	EMPLOYEE	1
SALES	MANAGER	1
SALES	SALESMAN	4

6.- [employees] Display the maximum number of employees of all the departments.

max_number
7

```
+------------+
```

7.- [employees] Show the name and the average salary of the departments whose average salary is greater than the average of salaries of all employees.

```
+------------+----------------+
| name       | average_salary |
+------------+----------------+
| ACCOUNTING |      2287.5000 |
| RESEARCH   |      2448.5714 |
+------------+----------------+
```

8.- [northwind] Show total cost of every order in May 1998 (also called “subtotal”). Clue: price*quantity*(1-discount)

```
+---------+---------------------+------------+
| OrderID | OrderDate           | subtotal   |
+---------+---------------------+------------+
|   11064 | 1998-05-01 00:00:00 |  9208.4850 |
|   11065 | 1998-05-01 00:00:00 |   492.4920 |
|   11066 | 1998-05-01 00:00:00 |  1811.0625 |
|   11067 | 1998-05-04 00:00:00 |   169.3575 |
|   11068 | 1998-05-04 00:00:00 |  4650.3600 |
|   11069 | 1998-05-04 00:00:00 |   702.0000 |
|   11070 | 1998-05-05 00:00:00 |  3653.3250 |
|   11071 | 1998-05-05 00:00:00 |   994.5000 |
|   11072 | 1998-05-05 00:00:00 | 10175.1000 |
|   11073 | 1998-05-05 00:00:00 |   585.0000 |
|   11074 | 1998-05-06 00:00:00 |   476.3850 |
|   11075 | 1998-05-06 00:00:00 |  1142.7000 |
|   11076 | 1998-05-06 00:00:00 |  2061.1500 |
|   11077 | 1998-05-06 00:00:00 |  2417.4930 |
+---------+---------------------+------------+
```

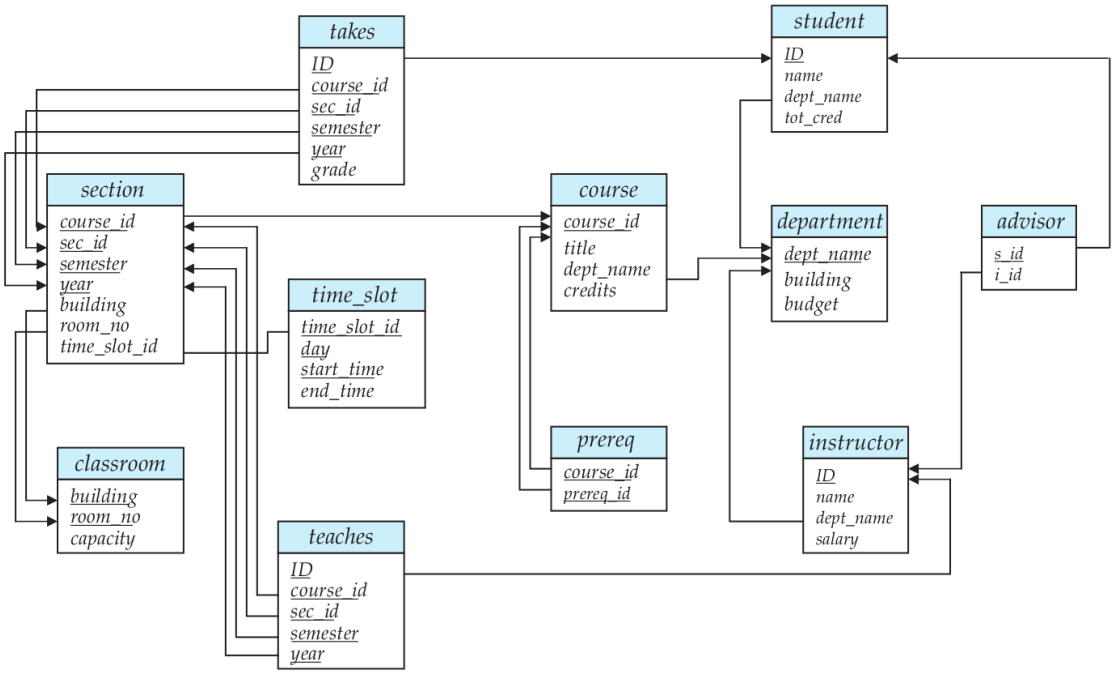
9.- [northwind] Show employee sales ordered by sales amount descendent.

```
+-----------+-----------+------------+-------------+
| LastName  | FirstName | EmployeeID | Subtotal    |
+-----------+-----------+------------+-------------+
| Peacock   | Margaret  |          4 | 487859.4435 |
| Leverling | Janet     |          3 | 415097.5140 |
| Davolio   | Nancy     |          1 | 393351.2700 |
| Fuller    | Andrew    |          2 | 346611.0570 |
| King      | Robert    |          7 | 275527.1805 |
| Callahan  | Laura     |          8 | 258721.0860 |
| Dodsworth | Anne      |          9 | 161779.8000 |
```

```
| Suyama     | Michael   |           6 | 152248.7850 |
| Buchanan   | Steven    |           5 | 147271.3125 |
+------------+-----------+-------------+-------------+
```

10.- [northwind] Top 5 products with more units sold.

```
+-------------------------+-------+
| ProductName             | Units |
+-------------------------+-------+
| Camembert Pierrot       |  1577 |
| Raclette Courdavault    |  1496 |
| Gorgonzola Telino       |  1397 |
| Gnocchi di nonna Alice  |  1263 |
| Pavlova                 |  1158 |
+-------------------------+-------+
```



DEPARTMENTS:

num	name	town_code
10	ACCOUNTING	SVQ
20	RESEARCH	MAD
30	SALES	BCN
40	PRODUCTION	BIO

EMPLOYEES:

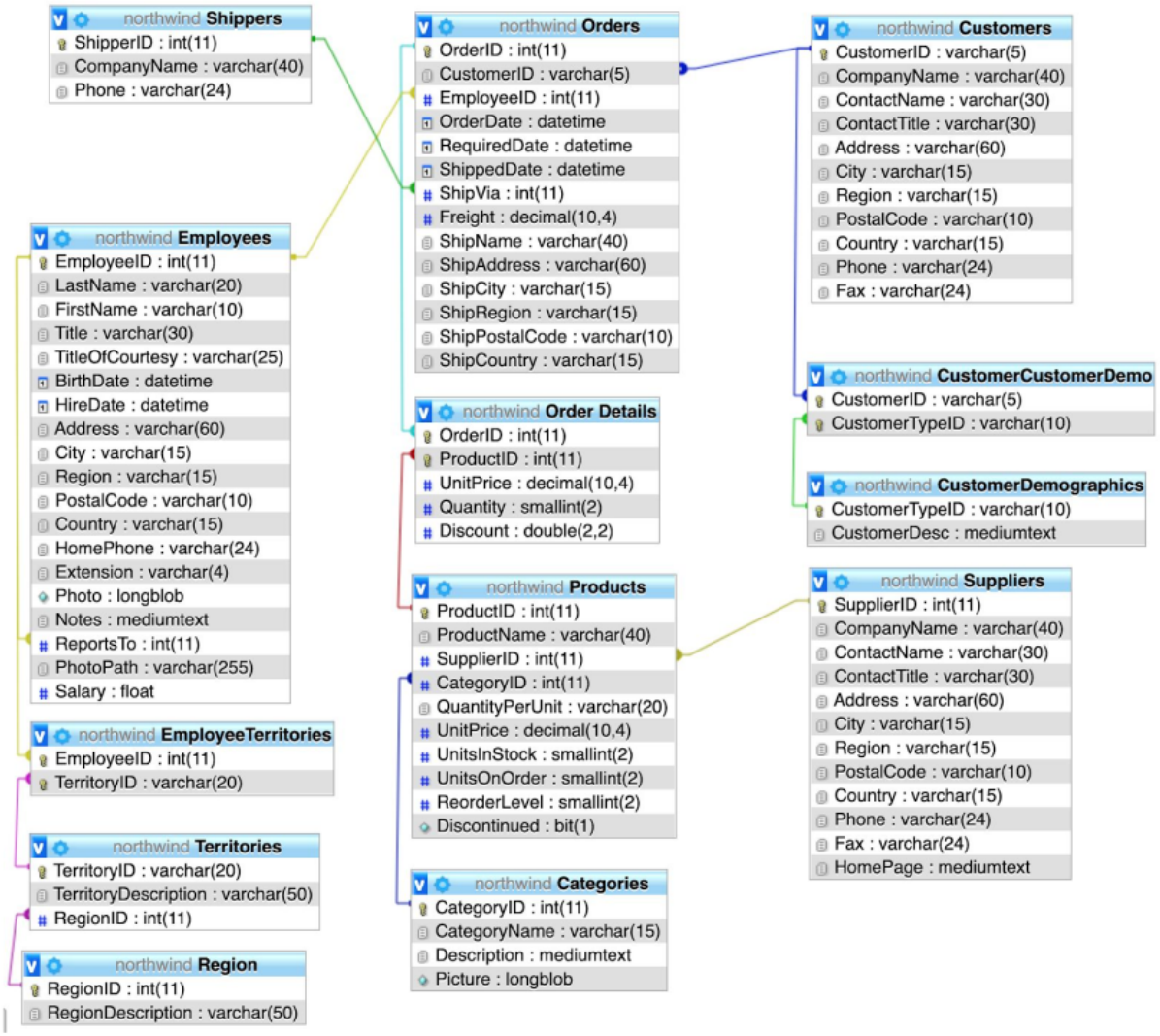
num	surname	name	manager	start_date	salary	commission	dept_num	occu_code
800	BANDERAS	ANTONIO	7839	1991-01-09	2885	NULL	20	MAN
7369	SÁNCHEZ	SERGIO	7902	1990-12-17	1040	NULL	20	EMP
7499	ARROYO	MARTA	7698	1990-02-20	1500	390	30	SAL
7521	SALA	RAUL	7698	1991-02-22	1625	650	30	SAL
7566	JIMÉNEZ	JUDIT	7839	1991-04-02	2900	NULL	20	MAN
7654	MARTÍN	MONICA	7698	1991-09-29	1600	1020	30	SAL
7698	NEGRO	BARTOLOME	7839	1991-05-01	3005	NULL	30	MAN
7782	CEREZO	ENRIQUE	7839	1991-06-09	2885	NULL	10	MAN
7788	GIL	JESUS	7566	1991-11-09	3000	NULL	20	ANA
7844	TOVAR	LUIS	7698	1991-09-08	1350	0	30	SAL
7876	ALONSO	FERNANDO	7788	1991-09-23	1430	NULL	20	EMP
7900	JIMENO	XAVIER	7698	1991-12-03	1335	NULL	30	EMP
7902	FERNÁNDEZ	ANA	7566	1991-12-03	3000	NULL	20	ANA
7934	MUÑOZ	ANTONIA	7782	1992-01-23	1690	NULL	10	EMP
8001	RUIZ	FERNANDA	7839	1992-06-10	2885	NULL	20	MAN

OCCUPATIONS:

code	name
ANA	ANALYST
EMP	EMPLOYEE
MAN	MANAGER
PRE	PRESIDENT
SAL	SALESMAN

TOWNS:

code	name
BCN	BARCELONA
BIO	BILBAO
MAD	MADRID
SVQ	SEVILLA

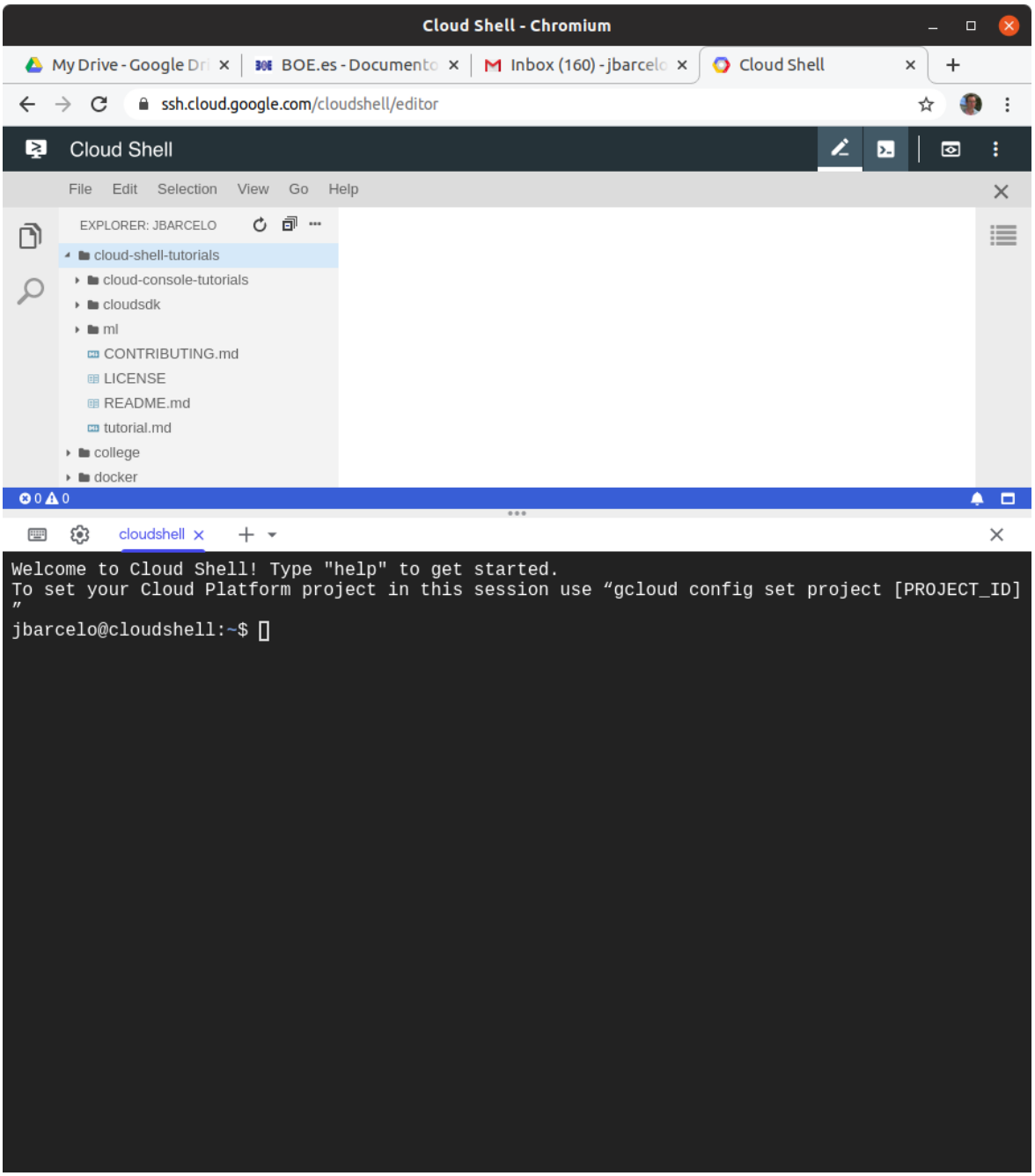

northwind Shippers
ShipperID : int(11)
CompanyName : varchar(40)
Phone : varchar(24)
northwind Orders
OrderID : int(11)
CustomerID : varchar(5)
EmployeeID : int(11)
OrderDate : datetime
RequiredDate : datetime
ShippedDate : datetime
ShipVia : int(11)
Freight : decimal(10,4)
ShipName : varchar(40)
ShipAddress : varchar(60)
ShipCity : varchar(15)
ShipRegion : varchar(15)
ShipPostalCode : varchar(10)
ShipCountry : varchar(15)
northwind Customers
CustomerID : varchar(5)
CompanyName : varchar(40)
ContactName : varchar(30)
ContactTitle : varchar(30)
Address : varchar(60)
City : varchar(15)
Region : varchar(15)
PostalCode : varchar(10)
Country : varchar(15)
Phone : varchar(24)
Fax : varchar(24)
northwind Employees
EmployeeID : int(11)
LastName : varchar(20)
FirstName : varchar(10)
Title : varchar(30)
TitleOfCourtesy : varchar(25)
BirthDate : datetime
HireDate : datetime
Address : varchar(60)
City : varchar(15)
Region : varchar(15)
PostalCode : varchar(10)
Country : varchar(15)
HomePhone : varchar(24)
Extension : varchar(4)
Photo : longblob
Notes : mediumtext
ReportsTo : int(11)
PhotoPath : varchar(255)
Salary : float
northwind CustomerCustomerDemo
CustomerID : varchar(5)
CustomerTypeID : varchar(10)
northwind Order Details
OrderID : int(11)
ProductID : int(11)
UnitPrice : decimal(10,4)
Quantity : smallint(2)
Discount : double(2,2)
northwind CustomerDemographics
CustomerTypeID : varchar(10)
CustomerDesc : mediumtext
northwind EmployeeTerritories
EmployeeID : int(11)
TerritoryID : varchar(20)
northwind Products
ProductID : int(11)
ProductName : varchar(40)
SupplierID : int(11)
CategoryID : int(11)
QuantityPerUnit : varchar(20)
UnitPrice : decimal(10,4)
UnitsInStock : smallint(2)
UnitsOnOrder : smallint(2)
ReorderLevel : smallint(2)
Discontinued : bit(1)
northwind Suppliers
SupplierID : int(11)
CompanyName : varchar(40)
ContactName : varchar(30)
ContactTitle : varchar(30)
Address : varchar(60)
City : varchar(15)
Region : varchar(15)
PostalCode : varchar(10)
Country : varchar(15)
Phone : varchar(24)
Fax : varchar(24)
HomePage : mediumtext
northwind Territories
TerritoryID : varchar(20)
TerritoryDescription : varchar(50)
RegionID : int(11)
northwind Categories
CategoryID : int(11)
CategoryName : varchar(15)
Description : mediumtext
Picture : longblob
northwind Region
RegionID : int(11)
RegionDescription : varchar(50)

5 Unit 5: Intermediate SQL 1

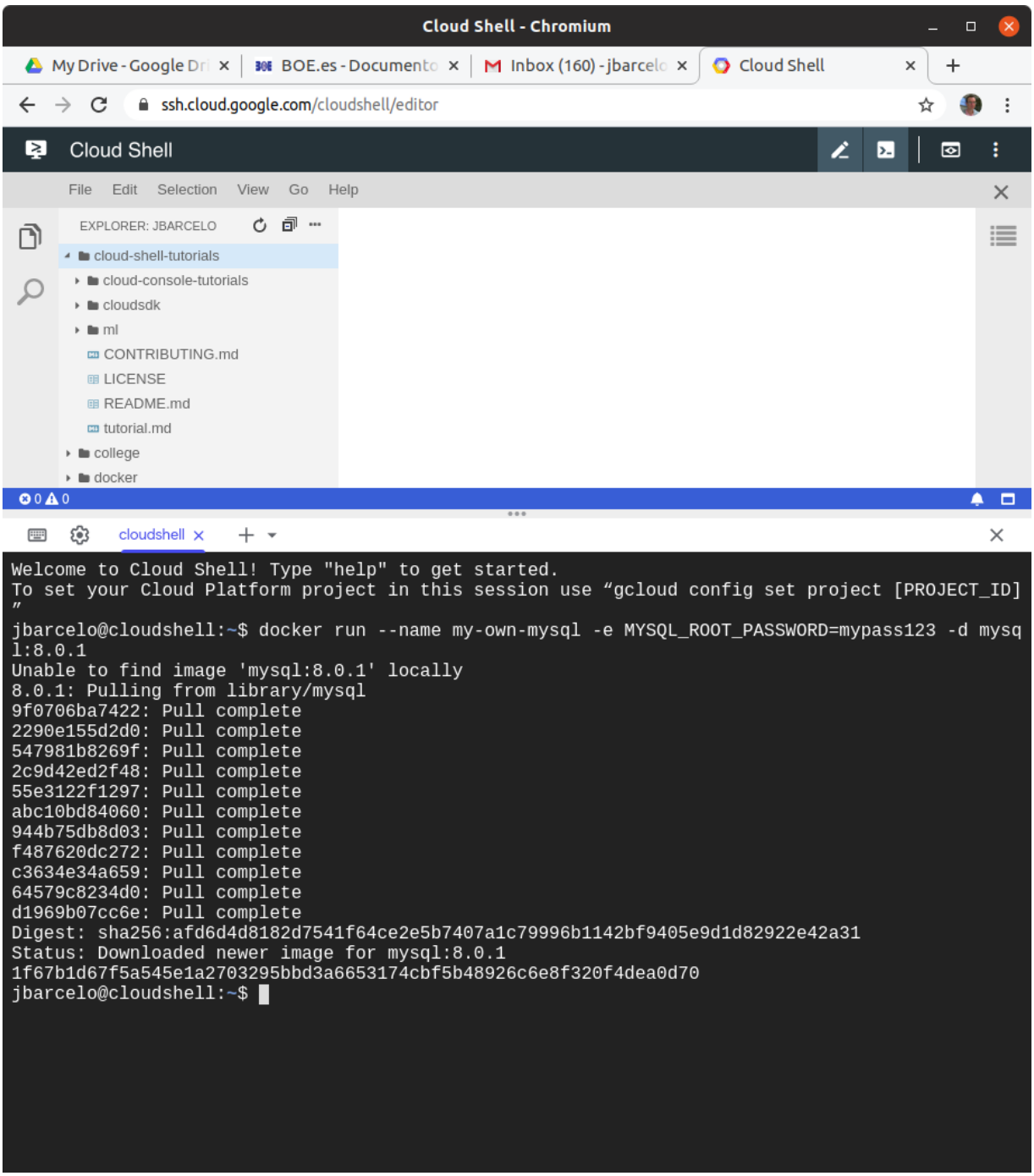
Course: Database management
Unit: Intermediate SQL 1
Assignment: Dockerized phpMyAdmin in Google CloudShell
Teamwork: Groups of two or three

Work in pairs and deliver a report of your activity. Complete the following tasks:

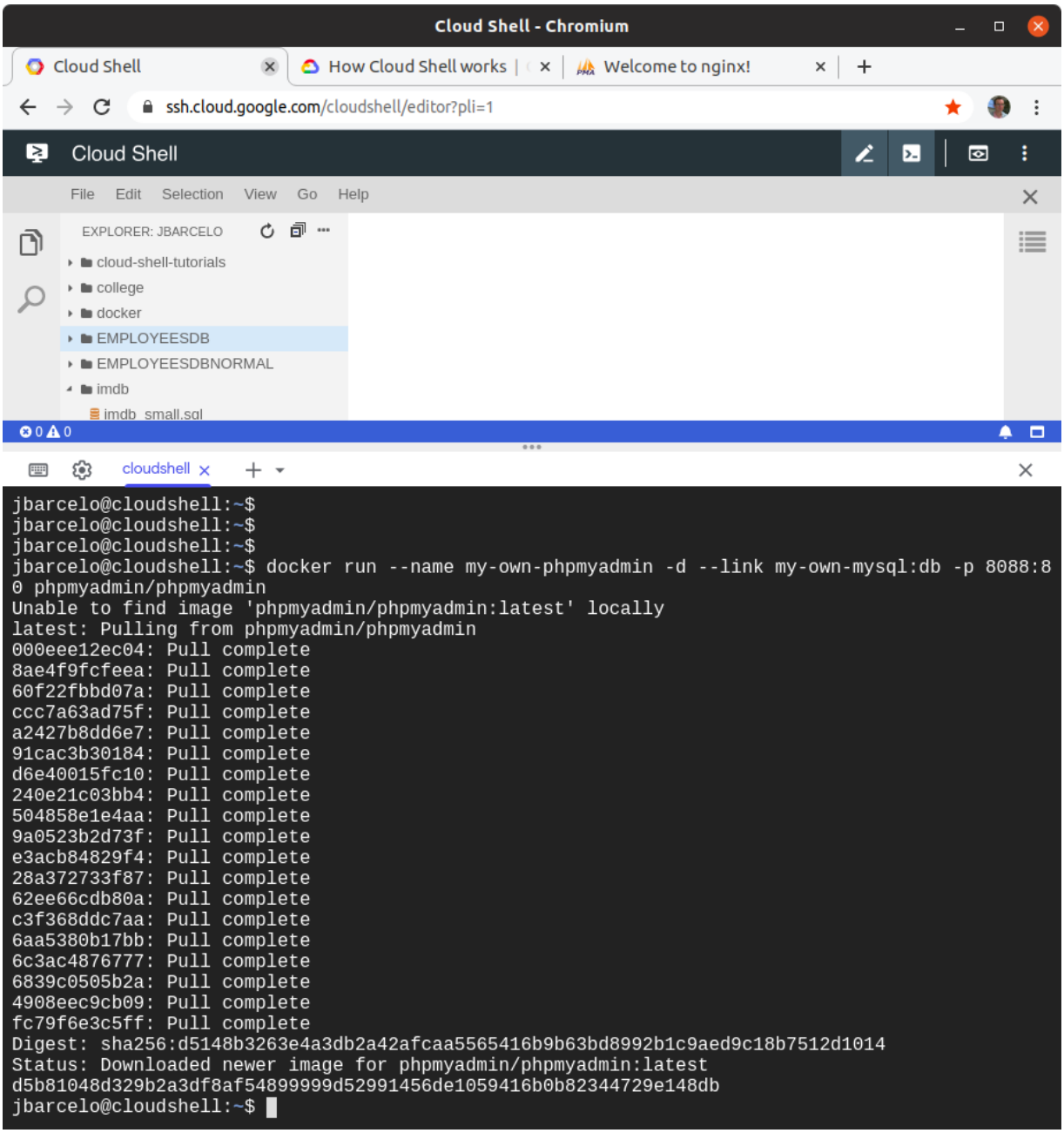
1 Open a google cloudshell.



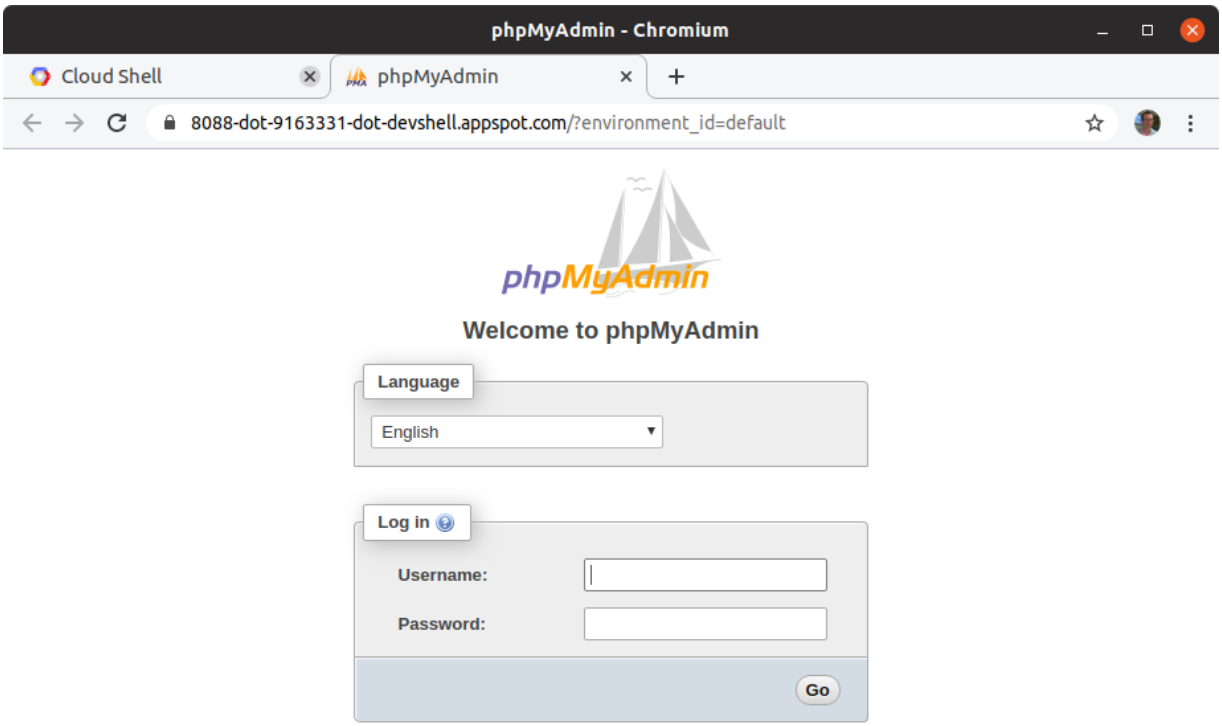
2 Launch a dockerized mysql database.



3 Launch a dockerized phpMyAdmin connected to the database of the previous step.



4 Use the “web preview” button to connect to phpMyAdmin



https://8088-dot-9163331-dot-devshell.appspot.com/url.php?url=https%3A%2F%2Fwww.phpmyadmin.net%2F

5 Create de university database. Create the instructor and department tables and fill them with data. Create the primary keys and the foreign key.

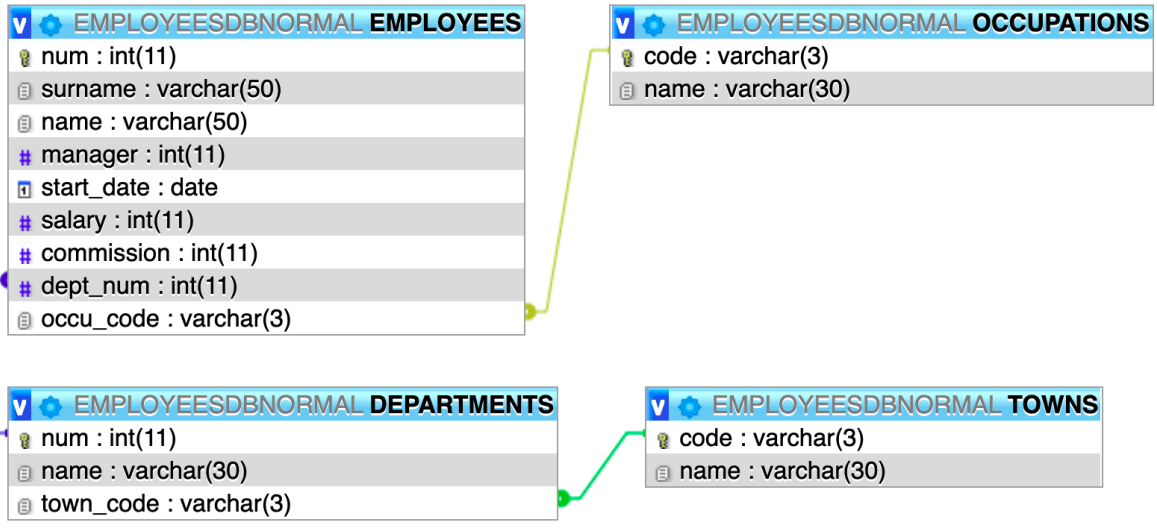
6 Provide feedback and suggestions regarding this assignment.

Course: Database management
Unit: Intermediate SQL 1
Material: Join

In order to have the same database, please import this database:

```
CREATE DATABASE IF NOT EXISTS `P04_queries01` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;
USE `P04_queries01`;

DROP TABLE IF EXISTS `DEPARTMENTS`;
CREATE TABLE `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `town_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `DEPARTMENTS` (`num`, `name`, `town_code`) VALUES
(10, 'ACCOUNTING', 'SVQ'),
(20, 'RESEARCH', 'MAD'),
(30, 'SALES', 'BCN'),
(40, 'PRODUCTION', 'BIO');

DROP TABLE IF EXISTS `EMPLOYEES`;
CREATE TABLE `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `manager` int(11) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL,
  `occu_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `manager`, `start_date`, `salary`,
`commission`, `dept_num`, `occu_code`) VALUES
(800, 'BANDERAS', 'ANTONIO', NULL, '1991-01-09', 2885, NULL, 20, NULL),
(7369, 'SÁNCHEZ', 'SERGIO', 7902, '1990-12-17', 1040, NULL, NULL, NULL),
(7499, 'ARROYO', 'MARTA', 7698, '1990-02-20', 1500, 390, 30, 'SAL'),
(7521, 'AGUILO', 'JOSEP', 7698, '1991-02-22', 1625, 650, 30, 'SAL'),
(7566, 'AROCA', 'JUDIT', 7839, '1991-04-02', 2900, NULL, 20, 'MAN'),
(7654, 'MARTÍN', 'MONICA', 7698, '1991-09-29', 1600, 1020, 30, 'SAL'),
(7698, 'AMER', 'BARTOLOME', 7839, '1991-05-01', 3005, NULL, 30, NULL),
(7782, 'COLOM', 'ENRIQUE', 7839, '1991-06-09', 2885, NULL, 10, 'MAN'),
(7788, 'GIL', 'JAVIER', 7566, '1991-11-09', 3000, NULL, 20, 'ANA'),
(7844, 'TOVAR', 'LUIS', 7698, '1991-09-08', 1350, 0, 30, 'SAL'),
(7876, 'ALONSO', 'FERNANDO', 7788, '1991-09-23', 1430, NULL, 20, 'EMP'),
(7900, 'JIMENO', 'XAVIER', 7698, '1991-12-03', 1335, NULL, 30, 'EMP'),
(7902, 'FERNÁNDEZ', 'ANA', 7566, '1991-12-03', 3000, NULL, NULL, 'ANA'),
(7934, 'MUÑOZ', 'ANTONIA', 7782, '1992-01-23', 1690, NULL, 10, 'EMP'),
(8001, 'RUIZ', 'FERNANDA', 7839, '1992-06-10', 2885, NULL, 20, 'MAN');

DROP TABLE IF EXISTS `OCCUPATIONS`;
CREATE TABLE `OCCUPATIONS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `OCCUPATIONS` (`code`, `name`) VALUES
```

```
('ANA', 'ANALYST'),
('EMP', 'EMPLOYEE'),
('MAN', 'MANAGER'),
('PRE', 'PRESIDENT'),
('SAL', 'SALESMAN');

DROP TABLE IF EXISTS `TOWNS`;
CREATE TABLE `TOWNS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `TOWNS` (`code`, `name`) VALUES
('BCN', 'BARCELONA'),
('BIO', 'BILBAO'),
('MAD', 'MADRID'),
('SVQ', 'SEVILLA');

ALTER TABLE `DEPARTMENTS`
  ADD PRIMARY KEY (`num`),
  ADD KEY `town_code` (`town_code`);

ALTER TABLE `EMPLOYEES`
  ADD PRIMARY KEY (`num`),
  ADD KEY `dept_num` (`dept_num`),
  ADD KEY `occu_code` (`occu_code`);

ALTER TABLE `OCCUPATIONS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `TOWNS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `DEPARTMENTS`
  ADD CONSTRAINT `DEPARTMENTS_ibfk_1` FOREIGN KEY (`town_code`) REFERENCES `TOWNS`
(`code`);

ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES `DEPARTMENTS`
(`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`occu_code`) REFERENCES `OCCUPATIONS`
(`code`);
COMMIT;
```



```
+------+------------+-----------+---------+------------+--------+------------+----------+-----------+
| num  | surname    | name      | manager | start_date | salary | commission | dept_num | occu_code |
+------+------------+-----------+---------+------------+--------+------------+----------+-----------+
|  800 | BANDERAS   | ANTONIO   |    NULL | 1991-01-09 |   2885 |       NULL |       20 | NULL      |
| 7369 | SÁNCHEZ    | SERGIO    |    7902 | 1990-12-17 |   1040 |       NULL |     NULL | NULL      |
| 7499 | ARROYO     | MARTA     |    7698 | 1990-02-20 |   1500 |        390 |       30 | SAL       |
| 7521 | AGUILO     | JOSEP     |    7698 | 1991-02-22 |   1625 |        650 |       30 | SAL       |
| 7566 | AROCA      | JUDIT     |    7839 | 1991-04-02 |   2900 |       NULL |       20 | MAN       |
| 7654 | MARTÍN     | MONICA    |    7698 | 1991-09-29 |   1600 |       1020 |       30 | SAL       |
| 7698 | AMER       | BARTOLOME |    7839 | 1991-05-01 |   3005 |       NULL |       30 | NULL      |
| 7782 | COLOM      | ENRIQUE   |    7839 | 1991-06-09 |   2885 |       NULL |       10 | MAN       |
| 7788 | GIL        | JAVIER    |    7566 | 1991-11-09 |   3000 |       NULL |       20 | ANA       |
| 7844 | TOVAR      | LUIS      |    7698 | 1991-09-08 |   1350 |          0 |       30 | SAL       |
| 7876 | ALONSO     | FERNANDO  |    7788 | 1991-09-23 |   1430 |       NULL |       20 | EMP       |
| 7900 | JIMENO     | XAVIER    |    7698 | 1991-12-03 |   1335 |       NULL |       30 | EMP       |
| 7902 | FERNÁNDEZ  | ANA       |    7566 | 1991-12-03 |   3000 |       NULL |     NULL | ANA       |
| 7934 | MUÑOZ      | ANTONIA   |    7782 | 1992-01-23 |   1690 |       NULL |       10 | EMP       |
| 8001 | RUIZ       | FERNANDA  |    7839 | 1992-06-10 |   2885 |       NULL |       20 | MAN       |
+------+------------+-----------+---------+------------+--------+------------+----------+-----------+
15 rows in set (0.001 sec)
```

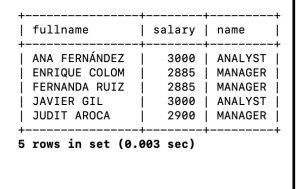
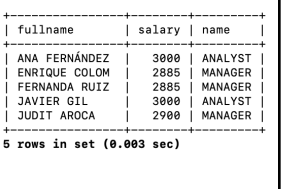
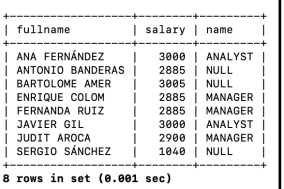
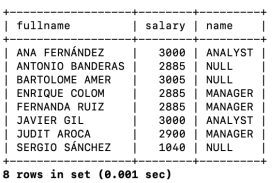
```
MariaDB [EMPLOYEESDBNORMAL]> select * from DEPARTMENTS;
+-----+------------+------------+
| num | name       | town_code  |
+-----+------------+------------+
|  10 | ACCOUNTING | SVQ        |
|  20 | RESEARCH   | MAD        |
|  30 | SALES      | BCN        |
|  40 | PRODUCTION | BIO        |
+-----+------------+------------+
4 rows in set (0.002 sec)
```

```
MariaDB [EMPLOYEESDBNORMAL]> select * from OCCUPATIONS;
+------+-----------+
| code | name      |
+------+-----------+
| ANA  | ANALYST   |
| EMP  | EMPLOYEE  |
| MAN  | MANAGER   |
| PRE  | PRESIDENT |
| SAL  | SALESMAN  |
+------+-----------+
5 rows in set (0.003 sec)
```

```
MariaDB [EMPLOYEESDBNORMAL]> select * from TOWNS;
+------+-----------+
| code | name      |
+------+-----------+
| BCN  | BARCELONA |
| BIO  | BILBAO    |
| MAD  | MADRID    |
| SVQ  | SEVILLA   |
+------+-----------+
4 rows in set (0.003 sec)
```

1. Show the full name of employees, salary and occupation name whose salary is not between 1100 and 2000. Sort the results by full name. Make four versions:

- Using an explicit inner join,
- using an implicit inner join,
- using a left outer join,
- and using a right outer join.

explicit inner join	implicit inner join	left outer join	right outer join
			

```
+-----------------+--------+---------+
| fullname        | salary | name    |
+-----------------+--------+---------+
| ANA FERNÁNDEZ   |   3000 | ANALYST |
| ENRIQUE COLOM   |   2885 | MANAGER |
| FERNANDA RUIZ   |   2885 | MANAGER |
| JAVIER GIL      |   3000 | ANALYST |
| JUDIT AROCA     |   2900 | MANAGER |
+-----------------+--------+---------+
5 rows in set (0.003 sec)
```

```
+-----------------+--------+---------+
| fullname        | salary | name    |
+-----------------+--------+---------+
| ANA FERNÁNDEZ   |   3000 | ANALYST |
| ENRIQUE COLOM   |   2885 | MANAGER |
| FERNANDA RUIZ   |   2885 | MANAGER |
| JAVIER GIL      |   3000 | ANALYST |
| JUDIT AROCA     |   2900 | MANAGER |
+-----------------+--------+---------+
5 rows in set (0.003 sec)
```

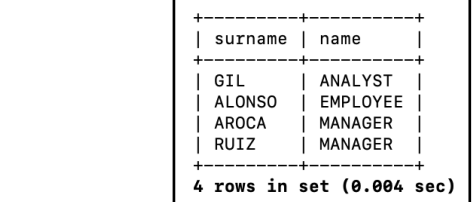
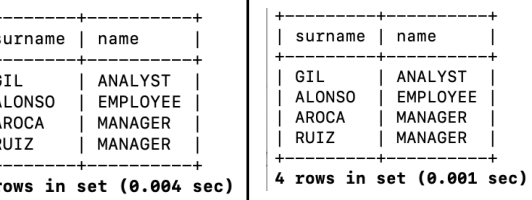
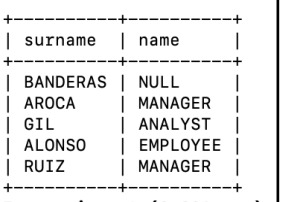
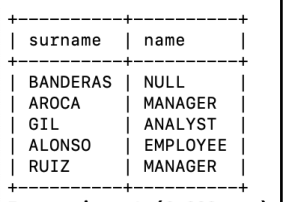
```
+-----------------+--------+---------+
| fullname        | salary | name    |
+-----------------+--------+---------+
| ANA FERNÁNDEZ   |   3000 | ANALYST |
| ANTONIO BANDERAS |  2885 | NULL    |
| BARTOLOME AMER  |   3005 | NULL    |
| ENRIQUE COLOM   |   2885 | MANAGER |
| FERNANDA RUIZ   |   2885 | MANAGER |
| JAVIER GIL      |   3000 | ANALYST |
| JUDIT AROCA     |   2900 | MANAGER |
| SERGIO SÁNCHEZ  |   1040 | NULL    |
+-----------------+--------+---------+
8 rows in set (0.001 sec)
```

```
+-----------------+--------+---------+
| fullname        | salary | name    |
+-----------------+--------+---------+
| ANA FERNÁNDEZ   |   3000 | ANALYST |
| ANTONIO BANDERAS |  2885 | NULL    |
| BARTOLOME AMER  |   3005 | NULL    |
| ENRIQUE COLOM   |   2885 | MANAGER |
| FERNANDA RUIZ   |   2885 | MANAGER |
| JAVIER GIL      |   3000 | ANALYST |
| JUDIT AROCA     |   2900 | MANAGER |
| SERGIO SÁNCHEZ  |   1040 | NULL    |
+-----------------+--------+---------+
8 rows in set (0.001 sec)
```

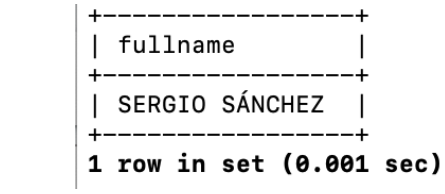
2. Select the surname and occupation of the employees of department number 20 (show occupation name instead occupation code). Make four versions:

- Using an explicit inner join,
- using an implicit inner join,
- using a left outer join,

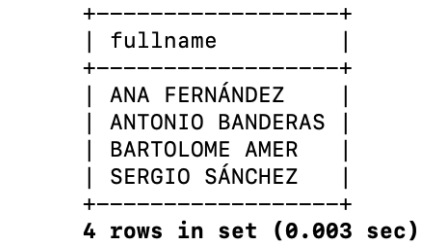
- and using a right outer join.

explicit inner join	implicit inner join	left outer join	right outer join
			

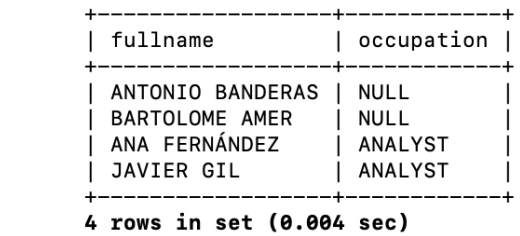
3. Show employee full name of the employees with no occupation and no department (sorted by full name).



4. Show employee full name of the employees with no occupation or no department (sorted by full name).



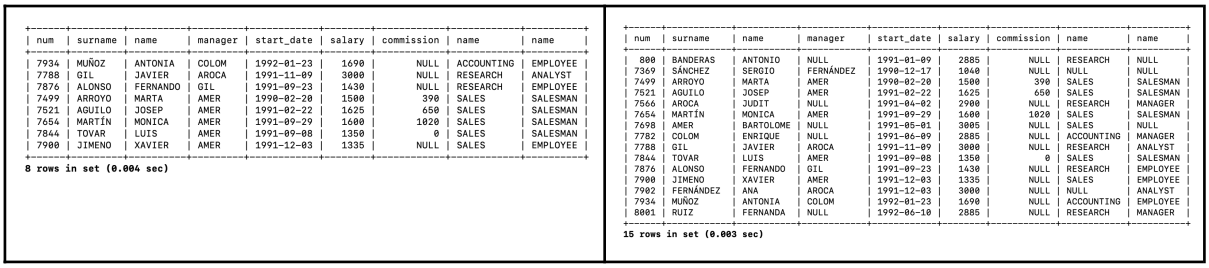
5. Show the full names of the employees amb occupation name whose occupation is neither "MANAGER" nor "EMPLOYEE" (=”occupation is not manager and is not employee”) and also have a salary higher than 2000.



6. Show all the data of all the employees (show department name instead department code, occupation name instead occupation code and manager surname instead manager num). Make two versions:

1. Using INNER JOIN.
2. Using LEFT OUTER JOIN.

INNER JOIN	OUTER JOIN



```
+------+---------+--------+---------+------------+--------+------------+------------+----------+
| num  | surname | name   | manager | start_date | salary | commission | name       | name     |
+------+---------+--------+---------+------------+--------+------------+------------+----------+
| 7934 | MUÑOZ   | ANTONIA | COLOM  | 1992-01-23 |   1690 |       NULL | ACCOUNTING | EMPLOYEE |
| 7788 | GIL     | JAVIER | AROCA   | 1991-11-09 |   3000 |       NULL | RESEARCH   | ANALYST  |
| 7876 | ALONSO  | FERNANDO | GIL   | 1991-09-23 |   1430 |       NULL | RESEARCH   | EMPLOYEE |
| 7499 | ARROYO  | MARTA  | AMER    | 1990-02-20 |   1500 |        390 | SALES      | SALESMAN |
| 7521 | AGUILO  | JOSEP  | AMER    | 1991-02-22 |   1625 |        650 | SALES      | SALESMAN |
| 7654 | MARTÍN  | MONICA | AMER    | 1991-09-29 |   1600 |       1020 | SALES      | SALESMAN |
| 7844 | TOVAR   | LUIS   | AMER    | 1991-09-08 |   1350 |          0 | SALES      | SALESMAN |
| 7900 | JIMENO  | XAVIER | AMER    | 1991-12-03 |   1335 |       NULL | SALES      | EMPLOYEE |
+------+---------+--------+---------+------------+--------+------------+------------+----------+
8 rows in set (0.004 sec)
```

```
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
| num  | surname | name      | manager | start_date | salary | commission | name       | name     |
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
|  800 | BANDERAS | ANTONIO  | NULL    | 1991-01-09 |   2885 |       NULL | RESEARCH   | NULL     |
| 7369 | SÁNCHEZ | SERGIO    | FERNÁNDEZ | 1990-12-17 |   1040 |       NULL | NULL       | NULL     |
| 7499 | ARROYO  | MARTA     | AMER    | 1990-02-20 |   1500 |        390 | SALES      | SALESMAN |
| 7521 | AGUILO  | JOSEP     | AMER    | 1991-02-22 |   1625 |        650 | SALES      | SALESMAN |
| 7566 | AROCA   | JUDIT     | NULL    | 1991-04-02 |   2900 |       NULL | RESEARCH   | MANAGER  |
| 7654 | MARTÍN  | MONICA    | AMER    | 1991-09-29 |   1600 |       1020 | SALES      | SALESMAN |
| 7698 | AMER    | BARTOLOME | NULL    | 1991-05-01 |   3005 |       NULL | SALES      | NULL     |
| 7782 | COLOM   | ENRIQUE   | NULL    | 1991-06-09 |   2885 |       NULL | ACCOUNTING | MANAGER  |
| 7788 | GIL     | JAVIER    | AROCA   | 1991-11-09 |   3000 |       NULL | RESEARCH   | ANALYST  |
| 7844 | TOVAR   | LUIS      | AMER    | 1991-09-08 |   1350 |          0 | SALES      | SALESMAN |
| 7876 | ALONSO  | FERNANDO  | GIL     | 1991-09-23 |   1430 |       NULL | RESEARCH   | EMPLOYEE |
| 7900 | JIMENO  | XAVIER    | AMER    | 1991-12-03 |   1335 |       NULL | SALES      | EMPLOYEE |
| 7902 | FERNÁNDEZ | ANA     | AROCA   | 1991-12-03 |   3000 |       NULL | NULL       | ANALYST  |
| 7934 | MUÑOZ   | ANTONIA   | COLOM   | 1992-01-23 |   1690 |       NULL | ACCOUNTING | EMPLOYEE |
| 8001 | RUIZ    | FERNANDA  | NULL    | 1992-06-10 |   2885 |       NULL | RESEARCH   | MANAGER  |
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
15 rows in set (0.003 sec)
```

7. Show the data of the employees whose salary is greater than 2000 (show department name instead department code, occupation name instead occupation code and manager surname instead manager num). Make two versions:

1. Using INNER JOIN.
2. Using LEFT OUTER JOIN.

If you think about how to do a third version using RIGHT OUTER JOIN you'll see that it's not possible...

INNER JOIN	OUTER JOIN

```
+------+---------+--------+---------+------------+--------+------------+----------+----------+
| num  | surname | name   | manager | start_date | salary | commission | name     | name     |
+------+---------+--------+---------+------------+--------+------------+----------+----------+
| 7788 | GIL     | JAVIER | AROCA   | 1991-11-09 |   3000 |       NULL | RESEARCH | ANALYST  |
+------+---------+--------+---------+------------+--------+------------+----------+----------+
1 row in set (0.003 sec)
```

```
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
| num  | surname | name      | manager | start_date | salary | commission | name       | name     |
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
|  800 | BANDERAS | ANTONIO  | NULL    | 1991-01-09 |   2885 |       NULL | RESEARCH   | NULL     |
| 7566 | AROCA   | JUDIT     | NULL    | 1991-04-02 |   2900 |       NULL | RESEARCH   | MANAGER  |
| 7698 | AMER    | BARTOLOME | NULL    | 1991-05-01 |   3005 |       NULL | SALES      | NULL     |
| 7782 | COLOM   | ENRIQUE   | NULL    | 1991-06-09 |   2885 |       NULL | ACCOUNTING | MANAGER  |
| 7788 | GIL     | JAVIER    | AROCA   | 1991-11-09 |   3000 |       NULL | RESEARCH   | ANALYST  |
| 7902 | FERNÁNDEZ | ANA     | AROCA   | 1991-12-03 |   3000 |       NULL | NULL       | ANALYST  |
| 8001 | RUIZ    | FERNANDA  | NULL    | 1992-06-10 |   2885 |       NULL | RESEARCH   | MANAGER  |
+------+---------+-----------+---------+------------+--------+------------+------------+----------+
7 rows in set (0.002 sec)
```

8.- Show number of employees per department considering employees with no department (clue: two queries with UNION).

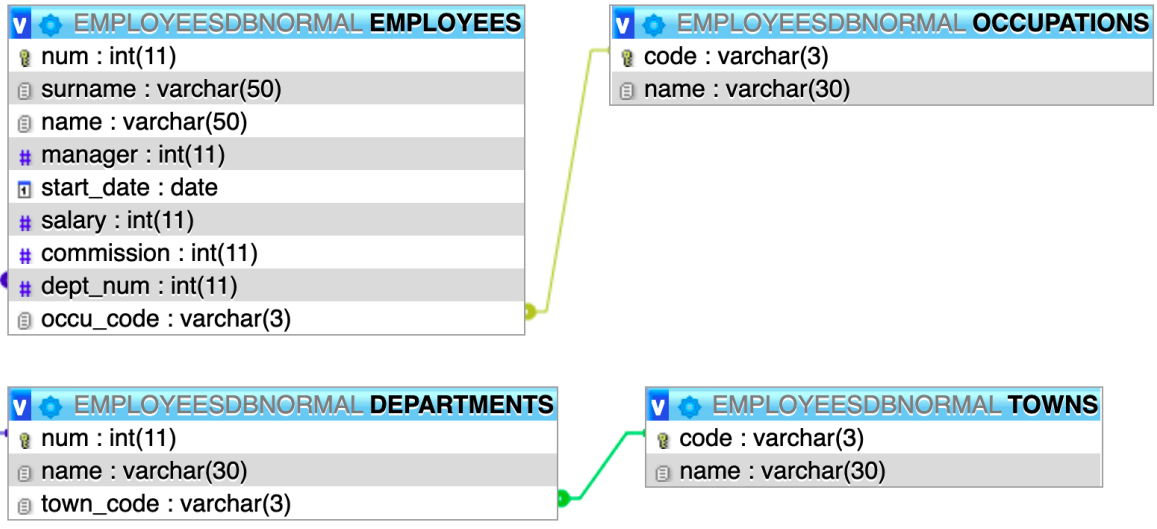
```
+------------+---------------+
| name       | num_employees |
+------------+---------------+
| ACCOUNTING |             2 |
| PRODUCTION |             0 |
| RESEARCH   |             5 |
| SALES      |             6 |
| NULL       |             2 |
+------------+---------------+
5 rows in set (0.002 sec)
```

Course: Database management
Unit: Intermediate SQL 1
Material: View

In order to have the same database, please import this database:

```
CREATE DATABASE IF NOT EXISTS `P04_views` DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;
USE `P04_views`;

DROP TABLE IF EXISTS `DEPARTMENTS`;
CREATE TABLE `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `town_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `DEPARTMENTS` (`num`, `name`, `town_code`) VALUES
(10, 'ACCOUNTING', 'SVQ'),
(20, 'RESEARCH', 'MAD'),
(30, 'SALES', 'BCN'),
(40, 'PRODUCTION', 'BIO');

DROP TABLE IF EXISTS `EMPLOYEES`;
CREATE TABLE `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `manager` int(11) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL,
  `occu_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `manager`, `start_date`, `salary`,
`commission`, `dept_num`, `occu_code`) VALUES
(800, 'BANDERAS', 'ANTONIO', NULL, '1991-01-09', 2885, NULL, 20, NULL),
(7369, 'SÁNCHEZ', 'SERGIO', 7902, '1990-12-17', 1040, NULL, NULL, NULL),
(7499, 'ARROYO', 'MARTA', 7698, '1990-02-20', 1500, 390, 30, 'SAL'),
(7521, 'AGUILO', 'JOSEP', 7698, '1991-02-22', 1625, 650, 30, 'SAL'),
(7566, 'AROCA', 'JUDIT', 7839, '1991-04-02', 2900, NULL, 20, 'MAN'),
(7654, 'MARTÍN', 'MONICA', 7698, '1991-09-29', 1600, 1020, 30, 'SAL'),
(7698, 'AMER', 'BARTOLOME', 7839, '1991-05-01', 3005, NULL, 30, NULL),
(7782, 'COLOM', 'ENRIQUE', 7839, '1991-06-09', 2885, NULL, 10, 'MAN'),
(7788, 'GIL', 'JAVIER', 7566, '1991-11-09', 3000, NULL, 20, 'ANA'),
(7844, 'TOVAR', 'LUIS', 7698, '1991-09-08', 1350, 0, 30, 'SAL'),
(7876, 'ALONSO', 'FERNANDO', 7788, '1991-09-23', 1430, NULL, 20, 'EMP'),
(7900, 'JIMENO', 'XAVIER', 7698, '1991-12-03', 1335, NULL, 30, 'EMP'),
(7902, 'FERNÁNDEZ', 'ANA', 7566, '1991-12-03', 3000, NULL, NULL, 'ANA'),
(7934, 'MUÑOZ', 'ANTONIA', 7782, '1992-01-23', 1690, NULL, 10, 'EMP'),
(8001, 'RUIZ', 'FERNANDA', 7839, '1992-06-10', 2885, NULL, 20, 'MAN');

DROP TABLE IF EXISTS `OCCUPATIONS`;
CREATE TABLE `OCCUPATIONS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `OCCUPATIONS` (`code`, `name`) VALUES
```

```
('ANA', 'ANALYST'),
('EMP', 'EMPLOYEE'),
('MAN', 'MANAGER'),
('PRE', 'PRESIDENT'),
('SAL', 'SALESMAN');

DROP TABLE IF EXISTS `TOWNS`;
CREATE TABLE `TOWNS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `TOWNS` (`code`, `name`) VALUES
('BCN', 'BARCELONA'),
('BIO', 'BILBAO'),
('MAD', 'MADRID'),
('SVQ', 'SEVILLA');

ALTER TABLE `DEPARTMENTS`
  ADD PRIMARY KEY (`num`),
  ADD KEY `town_code` (`town_code`);

ALTER TABLE `EMPLOYEES`
  ADD PRIMARY KEY (`num`),
  ADD KEY `dept_num` (`dept_num`),
  ADD KEY `occu_code` (`occu_code`);

ALTER TABLE `OCCUPATIONS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `TOWNS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `DEPARTMENTS`
  ADD CONSTRAINT `DEPARTMENTS_ibfk_1` FOREIGN KEY (`town_code`) REFERENCES `TOWNS`
(`code`);

ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES `DEPARTMENTS`
(`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`occu_code`) REFERENCES `OCCUPATIONS`
(`code`);
COMMIT;
```



1. Create a view (with name V_DEPARTMENTS) that shows all the departments with their number of employees and their town name.

```
MariaDB [P04_views]> select * from V_DEPARTMENTS;
+-----+------------+-----------+-----------+---------------+
| num | name       | town_code | town_name | num_employees |
+-----+------------+-----------+-----------+---------------+
|  10 | ACCOUNTING | SVQ       | SEVILLA   |             2 |
|  20 | RESEARCH   | MAD       | MADRID    |             5 |
|  30 | SALES      | BCN       | BARCELONA |             6 |
|  40 | PRODUCTION | BIO       | BILBAO    |             0 |
+-----+------------+-----------+-----------+---------------+
4 rows in set (0.001 sec)
```

2. Update the last view to add a field with the average salary of every department. To alter a view you can check this link: https://mariadb.com/kb/en/library/alter-view/

```
MariaDB [P04_views]> select * from V_DEPARTMENTS;
+-----+------------+-----------+-----------+---------------+----------------+
| num | name       | town_code | town_name | num_employees | average_salary |
+-----+------------+-----------+-----------+---------------+----------------+
|  10 | ACCOUNTING | SVQ       | SEVILLA   |             2 |      2287.5000 |
|  20 | RESEARCH   | MAD       | MADRID    |             5 |      2620.0000 |
|  30 | SALES      | BCN       | BARCELONA |             6 |      1735.8333 |
|  40 | PRODUCTION | BIO       | BILBAO    |             0 |           NULL |
+-----+------------+-----------+-----------+---------------+----------------+
4 rows in set (0.004 sec)
```

3. Insert a new departments (60 - 'HUMAN RESOURCES' - 'MAD') and a new employee with salary 9000€ who belongs to the new department (9999 - 'GONZALEZ' - 'SERGI' - NULL - '2019-01-01' - 9000 - NULL - 60 - NULL). Check if the data of the view changed automatically. Note that you must insert data into the tables EMPLOYEES and DEPARTMENTS (not into the view).

4. Create a view (with name V_EMPLOYEES) that shows all the employees with the name of their occupation name instead of the occupation code and the name of their department instead of the department number.

```
MariaDB [P04_views]> select * from V_EMPLOYEES;
+------+-----------+----------+---------+------------+--------+-----------------+-----------+
| num  | surname   | name     | manager | start_date | salary | dept_name       | occu_name |
+------+-----------+----------+---------+------------+--------+-----------------+-----------+
|  800 | BANDERAS  | ANTONIO  |    NULL | 1991-01-09 |   2885 | RESEARCH        | NULL      |
| 7369 | SÁNCHEZ   | SERGIO   |    7902 | 1990-12-17 |   1040 | NULL            | NULL      |
| 7499 | ARROYO    | MARTA    |    7698 | 1990-02-20 |   1500 | SALES           | SALESMAN  |
| 7521 | AGUILO    | JOSEP    |    7698 | 1991-02-22 |   1625 | SALES           | SALESMAN  |
| 7566 | AROCA     | JUDIT    |    7839 | 1991-04-02 |   2900 | RESEARCH        | MANAGER   |
| 7654 | MARTÍN    | MONICA   |    7698 | 1991-09-29 |   1600 | SALES           | SALESMAN  |
| 7698 | AMER      | BARTOLOME|    7839 | 1991-05-01 |   3005 | SALES           | NULL      |
| 7782 | COLOM     | ENRIQUE  |    7839 | 1991-06-09 |   2885 | ACCOUNTING      | MANAGER   |
| 7788 | GIL       | JAVIER   |    7566 | 1991-11-09 |   3000 | RESEARCH        | ANALYST   |
| 7844 | TOVAR     | LUIS     |    7698 | 1991-09-08 |   1350 | SALES           | SALESMAN  |
| 7876 | ALONSO    | FERNANDO |    7788 | 1991-09-23 |   1430 | RESEARCH        | EMPLOYEE  |
| 7900 | JIMENO    | XAVIER   |    7698 | 1991-12-03 |   1335 | SALES           | EMPLOYEE  |
| 7902 | FERNÁNDEZ | ANA      |    7566 | 1991-12-03 |   3000 | NULL            | ANALYST   |
| 7934 | MUÑOZ     | ANTONIA  |    7782 | 1992-01-23 |   1690 | ACCOUNTING      | EMPLOYEE  |
| 8001 | RUIZ      | FERNANDA |    7839 | 1992-06-10 |   2885 | RESEARCH        | MANAGER   |
| 9999 | GONZALEZ  | SERGI    |    NULL | 2019-01-01 |   9000 | HUMAN RESOURCES | NULL      |
+------+-----------+----------+---------+------------+--------+-----------------+-----------+
16 rows in set (0.001 sec)
```

5. Create a view (with name V_EMPLOYEES_FULL) using V_EMPLOYEES that shows manager's surname and name instead of manager num (you can NOT use the table EMPLOYEES you must use the view V_EMPLOYEES).

```
MariaDB [P04_views]> select * from V_EMPLOYEES_FULL;
+------+------------+-----------+---------+------------+--------+-----------------+-----------+-------------+-----------+
| num  | surname    | name      | manager | start_date | salary | dept_name       | occu_name | man_surname | man_name  |
+------+------------+-----------+---------+------------+--------+-----------------+-----------+-------------+-----------+
|  800 | BANDERAS   | ANTONIO   |    NULL | 1991-01-09 |   2885 | RESEARCH        | NULL      | NULL        | NULL      |
| 7369 | SÁNCHEZ    | SERGIO    |    7902 | 1990-12-17 |   1040 | NULL            | NULL      | FERNÁNDEZ   | ANA       |
| 7499 | ARROYO     | MARTA     |    7698 | 1990-02-20 |   1500 | SALES           | SALESMAN  | AMER        | BARTOLOME |
| 7521 | AGUILO     | JOSEP     |    7698 | 1991-02-22 |   1625 | SALES           | SALESMAN  | AMER        | BARTOLOME |
| 7566 | AROCA      | JUDIT     |    7839 | 1991-04-02 |   2900 | RESEARCH        | MANAGER   | NULL        | NULL      |
| 7654 | MARTÍN     | MONICA    |    7698 | 1991-09-29 |   1600 | SALES           | SALESMAN  | AMER        | BARTOLOME |
| 7698 | AMER       | BARTOLOME |    7839 | 1991-05-01 |   3005 | SALES           | NULL      | NULL        | NULL      |
| 7782 | COLOM      | ENRIQUE   |    7839 | 1991-06-09 |   2885 | ACCOUNTING      | MANAGER   | NULL        | NULL      |
| 7788 | GIL        | JAVIER    |    7566 | 1991-11-09 |   3000 | RESEARCH        | ANALYST   | AROCA       | JUDIT     |
| 7844 | TOVAR      | LUIS      |    7698 | 1991-09-08 |   1350 | SALES           | SALESMAN  | AMER        | BARTOLOME |
| 7876 | ALONSO     | FERNANDO  |    7788 | 1991-09-23 |   1430 | RESEARCH        | EMPLOYEE  | GIL         | JAVIER    |
| 7900 | JIMENO     | XAVIER    |    7698 | 1991-12-03 |   1335 | SALES           | EMPLOYEE  | AMER        | BARTOLOME |
| 7902 | FERNÁNDEZ  | ANA       |    7566 | 1991-12-03 |   3000 | NULL            | ANALYST   | AROCA       | JUDIT     |
| 7934 | MUÑOZ      | ANTONIA   |    7782 | 1992-01-23 |   1690 | ACCOUNTING      | EMPLOYEE  | COLOM       | ENRIQUE   |
| 8001 | RUIZ       | FERNANDA  |    7839 | 1992-06-10 |   2885 | RESEARCH        | MANAGER   | NULL        | NULL      |
| 9999 | GONZALEZ   | SERGI     |    NULL | 2019-01-01 |   9000 | HUMAN RESOURCES | NULL      | NULL        | NULL      |
+------+------------+-----------+---------+------------+--------+-----------------+-----------+-------------+-----------+
16 rows in set (0.001 sec)
```

6. Try to update the EMPLOYEE with number 9999 (for instance, try to change his/her surname) using the view V_EMPLOYEES. Did it work?

7. Now insert a new employee using the view V_EMPLOYEES. Did it work? Why?

Course: Database management
Unit: Intermediate SQL 1
Material: Transactions 1

In order to have the same database, please, run the following SQL code:

```
CREATE DATABASE `SQL1NORMALTRANSACTIONS`;
USE `SQL1NORMALTRANSACTIONS`;

CREATE TABLE `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `town_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `DEPARTMENTS` (`num`, `name`, `town_code`) VALUES
(10, 'ACCOUNTING', 'SVQ'),
(20, 'RESEARCH', 'MAD'),
(30, 'SALES', 'BCN'),
(40, 'PRODUCTION', 'BIO');

CREATE TABLE `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `manager` int(11) DEFAULT NULL,
  `begin_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL,
  `occu_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `manager`, `begin_date`, `salary`,
`commission`, `dept_num`, `occu_code`) VALUES
(1000, 'PITT', 'BRAD', NULL, '1984-01-01', 1040, NULL, 20, NULL),
(7369, 'SÁNCHEZ', 'SERGIO', 8001, '1990-12-17', 1040, NULL, 20, 'EMP'),
(7499, 'ARROYO', 'MARTA', 7698, '1990-02-20', 1500, 390, 30, 'SAL'),
(7521, 'SALA', 'RAUL', 7782, '1991-02-22', 1625, 650, 30, 'SAL'),
(7566, 'JIMÉNEZ', 'JUDIT', 1000, '1991-04-02', 2900, NULL, 20, 'MAN'),
(7654, 'MARTÍN', 'MONICA', 7698, '1991-09-29', 1600, 1020, 30, 'SAL'),
(7698, 'NEGRO', 'BARTOLOME', 1000, '1991-05-01', 3005, NULL, 30, 'MAN'),
(7782, 'CEREZO', 'ENRIQUE', 1000, '1991-06-09', 2885, NULL, 10, 'MAN'),
(7788, 'GIL', 'JESUS', 8000, '1991-11-09', 3000, NULL, 20, NULL),
(7844, 'TOVAR', 'LUIS', 7698, '1991-09-08', 1350, 0, 30, 'SAL'),
(7876, 'ALONSO', 'FERNANDO', 7788, '1991-09-23', 1430, NULL, 20, 'EMP'),
(7900, 'JIMENO', 'XAVIER', 8001, '1991-12-03', 1335, NULL, 30, 'EMP'),
(7902, 'FERNÁNDEZ', 'ANA', 8000, '1991-12-03', 3000, NULL, 20, NULL),
(7934, 'MUÑOZ', 'ANTONIA', 8001, '1992-01-23', 1690, NULL, 10, 'EMP'),
(8000, 'BANDERAS', 'ANTONIO', 1000, '1991-01-09', 2885, NULL, 20, 'MAN'),
(8001, 'RUIZ', 'FERNANDA', 1000, '1992-06-10', 2885, NULL, 20, 'MAN');

CREATE TABLE `OCCUPATIONS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `OCCUPATIONS` (`code`, `name`) VALUES
('ANA', 'ANALYST'),
('EMP', 'EMPLOYEE'),
('MAN', 'MANAGER'),
```

```
('PRE', 'PRESIDENT'),
('SAL', 'SALESMAN');

CREATE TABLE `TOWNS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `TOWNS` (`code`, `name`) VALUES
('BCN', 'BARCELONA'),
('BIO', 'BILBAO'),
('MAD', 'MADRID'),
('SVQ', 'SEVILLA');

ALTER TABLE `DEPARTMENTS`
  ADD PRIMARY KEY (`num`),
  ADD KEY `town_code` (`town_code`);

ALTER TABLE `EMPLOYEES`
  ADD PRIMARY KEY (`num`),
  ADD KEY `dept_num` (`dept_num`),
  ADD KEY `manager` (`manager`),
  ADD KEY `occu_code` (`occu_code`);

ALTER TABLE `OCCUPATIONS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `TOWNS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `DEPARTMENTS`
  ADD CONSTRAINT `DEPARTMENTS_ibfk_1` FOREIGN KEY (`town_code`) REFERENCES `TOWNS`
(`code`);

ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES `DEPARTMENTS`
(`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`manager`) REFERENCES `EMPLOYEES` (`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_3` FOREIGN KEY (`occu_code`) REFERENCES `OCCUPATIONS`
(`code`);
```

1. Connect to the VM on your VPS using the linux shell.

2. Use the database ‘SQL1_NORMAL’.

3. Begin a transaction (also set autocommit to off before beginning the transaction).

SET AUTOCOMMIT=0;
START TRANSACTION;

4. Query the average commission of the employees.

5. Increase employees’ commission by 20%.

6. Query again the average commission of the employees. Is it higher?

7. Delete the employees with surname “TOVAR”.

8. Rollback all the operations.

9. Query again the average commission of the employees and check if there are employees with surname “TOVAR”.

10. Create a savepoint with name “SPINCCOMMISSION”. (clue: note the start transaction again!)

11. Increase employees’ commission by 20%.

12. Query again the average commission of the employees. is it higher?

13. Create a savepoint with name “SPDELTOVAR”.

14. Delete the employees with surname “TOVAR”.

15. Rollback to savepoint “SPDELTOVAR”.

16. Query again the average commission of the employees and check if there are employees with surname “TOVAR”.

17. Commit all the operations.

18. Do employees with surname “TOVAR” exist or not? Is the commission increased?

Course: Database management
Unit: Intermediate SQL 1
Material: Transactions 2

Note: You need to experiment and verify yourself with the services of your DBMS. A concrete DBMS may differ in the way it support basic SQL transaction services.

1. Login into MariaDB as root and create a database ‘TestDB’ (to login as root you must be root in the shell prompt).

2. Create a new user ‘student’ (password ‘alualualu’) and grant all privileges to TestDB in localhost.

```
MariaDB [(none)]> GRANT ALL ON TestDB.* TO 'student'@'localhost'  IDENTIFIED BY 'alualualu';
--we’ll see how to create users later…
```

3. Exit from the "root" user session from the MariaDB session, and exit the root's session from bash.

4. Login to MariaDB as ‘student’ and use ‘TestDB’.

5. Create a new table named "T", having three columns: id (of type integer, the primary key), s (of type character string with a length varying from 1 to 40 characters), and si (of type small integer):

6. Check if the table was created with the command ‘DESCRIBE T’ and show the SQL code of the creation of the table with ‘SHOW CREATE TABLE T’.

NOTE: MariaDB in Linux is case insensitive, with the exception of table and database names. This means that the following work fine: "Describe T", "describe T", "create TaBle T ...", but "use testDB", and "describe t" will fail.

7. Insert the next rows into the table ‘T’. After that, check if the rows were really inserted.

1	‘first’
2	‘second’
3	‘third’

8. Execute a ‘ROLLBACK’. Did the changes disappeared? Why?

9. Delete the data inserted into T with a DELETE command. After that start a transaction and repeat exercises 7 and 8. Now, ROLLBACK the transaction. DML commands were rolled back?

10. Execute:

```
INSERT INTO T (id, s) VALUES (4, 'fourth');
ROLLBACK;
```

SELECT * FROM T;

- What's the result? Why?

11. Now, set autocommit to off.

12. Delete all the rows from T and execute 'COMMIT'.

13. Insert the next columns and execute a query to check if they were inserted.

5	'fifth'
6	'sixth'

14. Now execute 'ROLLBACK' and execute a query to show all the rows in the relation T. Are there the rows that you just inserted in exercise 13.

15. Execute:

```
SET AUTOCOMMIT=0;
INSERT INTO T (id, s) VALUES (9, 'will this be committed?');
CREATE TABLE T2 (id INT);
INSERT INTO T2 (id) VALUES (1);
SELECT * FROM T2;
ROLLBACK;
DROP TABLE T2;
COMMIT;
```

Secondly, execute:

```
SELECT * FROM T;
DESCRIBE T2;
```

Explain what happened.

16. Empty the contents of the table T (and check it!) with the following commands:

```
SET AUTOCOMMIT=0;
DELETE FROM T;
COMMIT;
SELECT * FROM T;
```

We will see this better in following units, but we'll use the following commands:

- SHOW ERRORS
- SHOW WARNINGS

Execute the following commands:

```
SET AUTOCOMMIT=0;
INSERT INTO T (id, s) VALUES (1, 'This row is OK, but errors start here...');
SHOW ERRORS;
SHOW WARNINGS;
-- Is this a mistake/warning?
SELECT (1/0) AS dummy FROM DUAL;
SHOW ERRORS;
SHOW WARNINGS;
```

```
-- Is this a mistake/warning?
UPDATE T SET s = 'foo' WHERE id = 9999 ;
SHOW ERRORS;
SHOW WARNINGS;
-- Is this a mistake/warning?
DELETE FROM T WHERE id = 7777 ;
SHOW ERRORS;
SHOW WARNINGS;
-- Is this a mistake/warning?
INSERT INTO T (id, s) VALUES (1, 'Hi, I am a duplicate');
SHOW ERRORS;
SHOW WARNINGS;
-- Is this a mistake/warning?
INSERT INTO T (id, s)
 VALUES (3, 'How about inserting too long of a string value?');
SHOW ERRORS;
SHOW WARNINGS;
-- Is this a mistake/warning?
INSERT INTO T (id, s, si) VALUES (4, 'Smallint overflow for 32769?', 32769);
SHOW ERRORS;
SHOW WARNINGS;
```

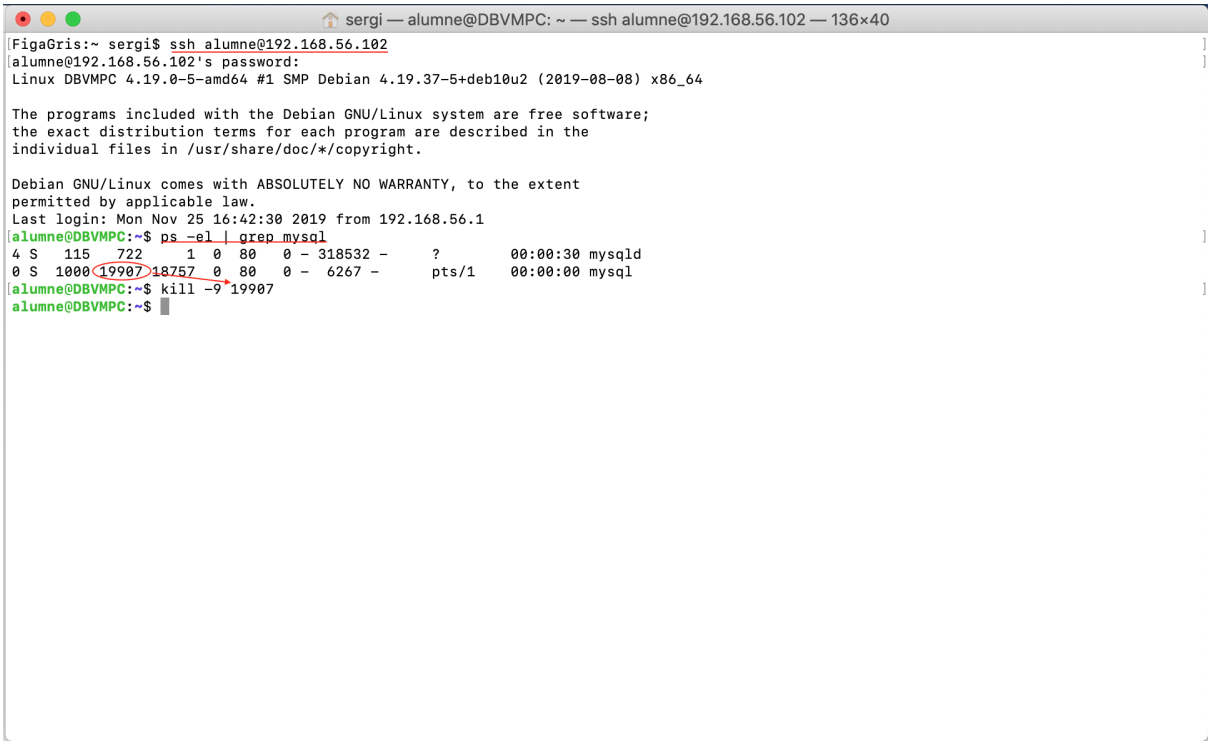
QUESTION: Explain every DML command (INSERT/UPDATE/DELETE) above and the reason why there is an error/warning or not.

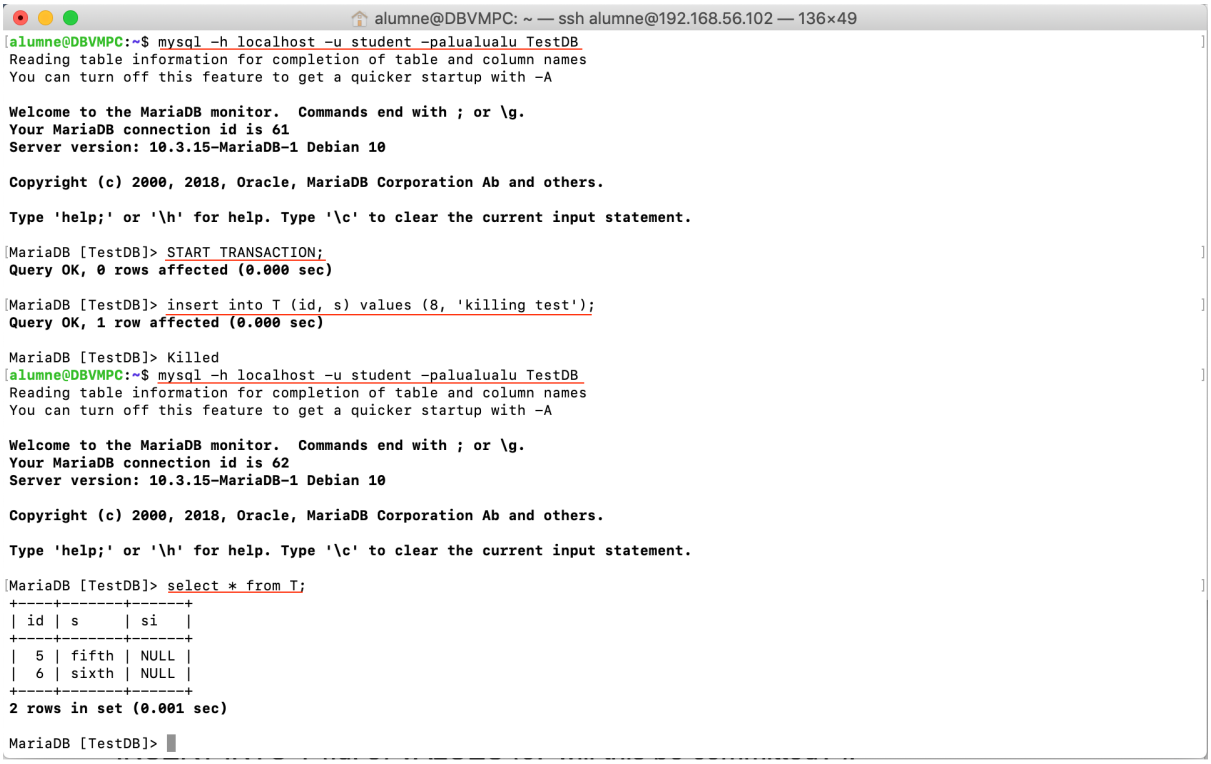
17. Execute:

```
INSERT INTO T (id, s) VALUES (5, 'Is the transaction still active?');
SELECT * FROM T;
```

QUESTION: Is the transaction still active? Explain your answer.

18. Begin a transaction with ‘START TRANSACTION’. After that insert a new row. In another terminal kill the process running mysql. Explain what happened with the file inserted.


sergi — alumne@DBVMPC: ~ — ssh alumne@192.168.56.102 — 136×40
FigaGris:~ sergi$ ssh alumne@192.168.56.102
alumne@192.168.56.102's password:
Linux DBVMPC 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5+deb10u2 (2019-08-08) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 25 16:42:30 2019 from 192.168.56.1
alumne@DBVMPC:~$ ps -el | grep mysql
4 S 115 722 1 0 80 0 - 318532 - ? 00:00:30 mysqld
0 S 1000 19907 18757 0 80 0 - 6267 - pts/1 00:00:00 mysql
alumne@DBVMPC:~$ kill -9 19907
alumne@DBVMPC:~$


alumne@DBVMPC: ~ — ssh alumne@192.168.56.102 — 136×49
alumne@DBVMPC:~$ mysql -h localhost -u student -palualualu TestDB
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 61
Server version: 10.3.15-MariaDB-1 Debian 10
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [TestDB]> START TRANSACTION;
Query OK, 0 rows affected (0.000 sec)
MariaDB [TestDB]> insert into T (id, s) values (8, 'killing test');
Query OK, 1 row affected (0.000 sec)
MariaDB [TestDB]> Killed
alumne@DBVMPC:~$ mysql -h localhost -u student -palualualu TestDB
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 62
Server version: 10.3.15-MariaDB-1 Debian 10
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [TestDB]> select * from T;
+----+-------+------+
| id | s | si |
+----+-------+------+
| 5 | fifth | NULL |
| 6 | sixth | NULL |
+----+-------+------+
2 rows in set (0.001 sec)
MariaDB [TestDB]>

Course: Database management
Unit: Intermediate SQL 1
Assignment: Two-phase locking and snapshot isolation
Teamwork: Groups of two or three

Discuss the importance of locking mechanisms to prevent that simultaneous modifications of the database result in an inconsistent state.

Think about examples of such scenario.

Mention two techniques to prevent overlapping modifications.

- Two-phase locking
- Snapshot isolation

Test lesson 5. Intermediate SQL 1

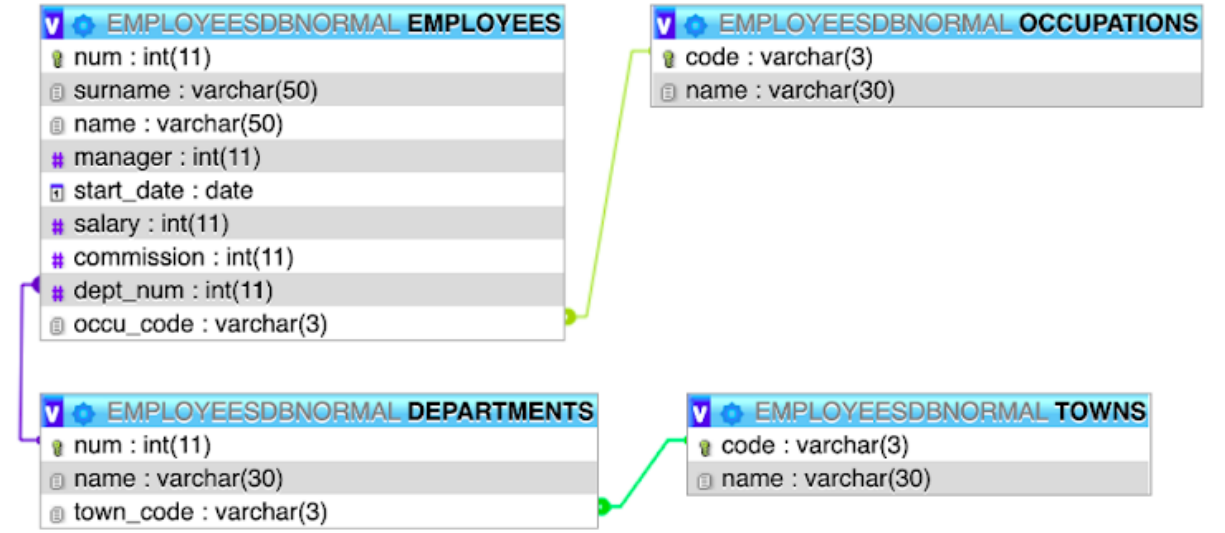
The first 5 correct answers add nothing to the total result. After that, each correct answer adds 6.66 points to the final mark. Wrong answers do not reduce the final mark. It is recommended that you answer all the questions.

The respondent's email (**null**) was recorded on submission of this form.

1. Email *

2. Show number of employees per department considering employees with no department

```
+------------+---------------+
| name       | num_employees |
+------------+---------------+
| ACCOUNTING |             2 |
| PRODUCTION |             0 |
| RESEARCH   |             5 |
| SALES      |             6 |
| NULL       |             2 |
+------------+---------------+
5 rows in set (0.002 sec)
```



Mark only one oval.

- ( ) select D.name, count(E.num) as num_employees from EMPLOYEES as E left join DEPARTMENTS as D on E.dept_num=D.num group by D.name;
- ( ) select D.name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num group by D.name;
- ( ) (select D.name, count(E.num) as num_employees from EMPLOYEES as E left join DEPARTMENTS as D on E.dept_num=D.num group by D.name) union (select D.name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num group by D.name);
- ( ) (select D.name, count(E.num) as num_employees from EMPLOYEES as E natural left join DEPARTMENTS as D on E.dept_num=D.num group by D.name) union (select D.name, count(E.num) as num_employees from EMPLOYEES as E natural right join DEPARTMENTS as D on E.dept_num=D.num group by D.name);

3. Given the tables course and prereq, the resulting table shown below is ...

course

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

prereq

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

course_id	*title*	*dept_name*	*credits*	*prereq_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

Mark only one oval.

- ◯ course natural inner join prereq
- ◯ course natural outer join prereq
- ◯ course natural right outer join prereq
- ◯ course natural left outer join prereq

4. [Join type – defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition) are treated.] Which of the following is not a join type?

Mark only one oval.

- ◯ natural join
- ◯ inner join
- ◯ left join
- ◯ right join

5. Which of the following is not implemented in MariaDB?

Mark only one oval.

- inner join
- full outer join
- left join
- right join

6. Imagine that the university clerk needs access to the name of the instructors but not their salaries. How can we filter the information of the instructor table.

Mark only one oval.

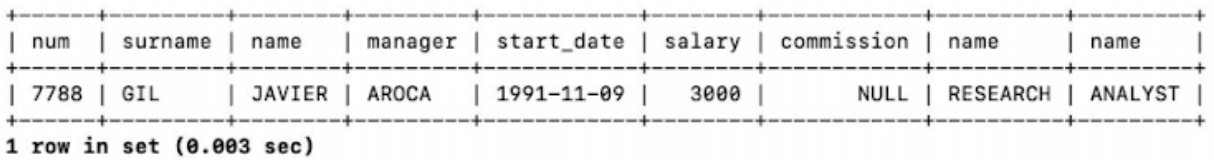
- view
- join
- check
- transaction

7. Which of the following statements is false?

Mark only one oval.

- The sintax to create a view is "CREATE VIEW v AS <query expression>".
- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.
- View definition is the same as creating a new relation by evaluating the query expression. The data in the view is not changed even if the data in the original tables is.
- A view definition causes the saving of an expression; the expression is substituted into queries using the view.

8. Show the data of the employees whose salary is greater than 2000 (show department name instead department code, occupation name instead occupation code and managersurname instead manager num) using inner join.



num	surname	name	manager	start_date	salary	commission	name	name
7788	GIL	JAVIER	AROCA	1991-11-09	3000	NULL	RESEARCH	ANALYST

1 row in set (0.003 sec)

Mark only one oval.

- select E.num, E.surname, E.name, M.surname, E.start_date, E.salary, E.commission, D.name, O.namefrom EMPLOYEES as E join DEPARTMENTS as D on E.dept_num=D.num join OCCUPATIONS as O on E.occu_code=O.code join EMPLOYEES as M on E.manager=M.num where E.salary>2000;
- select E.num, E.surname, E.name, M.surname, E.start_date, E.salary, E.commission, D.name, O.namefrom EMPLOYEES as E join DEPARTMENTS as D on E.dept_num=D.num join OCCUPATIONS as O on E.occu_code=O.code where E.salary>2000;
- select E.num, E.surname, E.name, T.name, E.start_date, E.salary, E.commission, D.name, O.name from EMPLOYEES as E join DEPARTMENTS as D on E.dept_num=D.num join OCCUPATIONS as O on E.occu_code=O.code join EMPLOYEES as M on E.manager=M.num join TOWNS as T on D.towncode=T.code where E.salary>2000;
- select E.num, E.surname, E.name, T.name, E.start_date, E.salary, E.commission, D.name, O.name from EMPLOYEES as E join DEPARTMENTS as D on E.dept_num=D.num join OCCUPATIONS as O on E.occu_code=O.code join TOWNS as T on D.towncode=T.code where E.salary>2000;

9. What command can we use to see the views and the tables?

Mark only one oval.

- show full tables;
- describe tables;
- show views;
- select * from views;

10. Create a view (with name V_DEPARTMENTS) that shows all the departments with their number of employees and their town name.

```
MariaDB [P04_views]> select * from V_DEPARTMENTS;
+-----+------------+-----------+-----------+---------------+
| num | name       | town_code | town_name | num_employees |
+-----+------------+-----------+-----------+---------------+
|  10 | ACCOUNTING | SVQ       | SEVILLA   |             2 |
|  20 | RESEARCH   | MAD       | MADRID    |             5 |
|  30 | SALES      | BCN       | BARCELONA |             6 |
|  40 | PRODUCTION | BIO       | BILBAO    |             0 |
+-----+------------+-----------+-----------+---------------+
4 rows in set (0.001 sec)
```

Mark only one oval.

- ◯ create view V_DEPARTMENTS select D.num, D.name, D.town_code, T.name as town_name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num join TOWNS as T on D.town_code=T.code group by D.num;
- ◯ create V_DEPARTMENTS as select D.num, D.name, D.town_code, T.name as town_name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num join TOWNS as T on D.town_code=T.code group by D.num;
- ◯ create view V_DEPARTMENTS as select D.num, D.name, D.town_code, T.name as town_name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num join TOWNS as T on D.town_code=T.code group by D.num;
- ◯ new view V_DEPARTMENTS as select D.num, D.name, D.town_code, T.name as town_name, count(E.num) as num_employees from EMPLOYEES as E right join DEPARTMENTS as D on E.dept_num=D.num join TOWNS as T on D.town_code=T.code group by D.num;

11. Which of the following statements is false?

Mark only one oval.

- ◯ It is possible to alter a view.
- ◯ It is possible to define a view selecting data from another view. "CREATE VIEW v2 AS SELECT column1 FROM v1;"
- ◯ It is possible to create views with columns that are not present in the original table. Like the following example "create view V_DEPARTMENTS as select D.num, D.name, T.code, T.name, count(E.num) from EMPLOYEES as E join DEPARTMENTS as D on E.dept_num=D.num join TOWNS as T on D.town_code=T.code group by D.num;"
- ◯ It is always possible to insert data into views.

12. What is the statement to start a transaction in MariaDB?

Mark only one oval.

- START TRANSACTION;
- RECORD TRANSACTION;
- SAVE CHECKPOINT;
- ADD CHECKPOINT;

13. What is the result or running the following statements? start transaction;delete from instructor where dept_name in (select dept_name from department where building = 'Watson');rollback;commit;

Mark only one oval.

- Nothing.
- It deletes the instructors of departments in the Watson building.
- It deletes the departments in the Watson building.
- It deletes the department name of the instructors that work in the Watson building.

14. What is the result or running the following statements? start transaction;delete from instructor where dept_name in (select dept_name from department where building = 'Watson'); savepoint my_savepoint; rollback to my_savepoint; commit;

Mark only one oval.

- Nothing.
- It deletes the instructors of departments in the Watson building.
- It deletes the departments in the Watson building.
- It deletes the department name of the instructors that work in the Watson building.

15. What is the keyword to finish a transaction in MariaDB?

Mark only one oval.

- Commit;
- Finish;
- End;
- Complete;

16. Which of the following is not an integrity constraint on a single relation?

Mark only one oval.

- rollback
- not null
- primary key
- unique

17. If we add the restriction "unique(building, budget)" to the "department" table of the "university" database, which of the following statements is false?

Mark only one oval.

- it is not possible to have two departments in the same building.
- the combination of attributes (dept_name, building) will be a superkey.
- it is not possible to have two departments in the same building with the same budget.
- the values of building and budget can be null.

18. Which of the following is not valid?

Mark only one oval.

- check(select name from instructor where dept_name in (select dept_name from department where building='Watson'))
- check (budget > 0)
- check (semester in ('Fall', 'Winter', 'Spring', 'Summer'))
- check (year > 1701 and year < 2100)

19. How can we add a constraint to an existing table "table-name"?

Mark only one oval.

- ALTER TABLE "table-name" ADD "constraint"
- CREATE "constraint" IN TABLE "table-name"
- DEFINE "table-name" NEW "constraint"
- ADD "constraint" TO TABLE "table-name"

20. How can we disable autocommit in MySQL/MariaDB?

Mark only one oval.

- set autocommit=0;
- set autocommit=1;
- set autocommit off;
- set autcommit on;

21. In the slides we have defined three "Join Conditions". Which of the following is not a "Join Coindition"?

Mark only one oval.

- OUTER
- NATURAL
- ON <predicate>
- USING (A1, A2, ..., An)

22. Consider the following code: INSERT INTO T (id, s) VALUES (4, 'fourth');ROLLBACK;SELECT * FROM T; What is the result?

Mark only one oval.

- The data will be inserted if autocommit is on.
- The data will be inserted if autocommit is off.
- The data will be inserted if there was a START TRANSACTION; code before.
- The data will be inserted if there was a BEGIN; code before.

23. 1. CREATE TABLE my_table (column1 INTEGER, column2 INTEGER, CONSTRAINT my_constraint UNIQUE (column1, column2)); 2. INSERT INTO my_table VALUES (1,2); 3. INSERT INTO my_table VALUES (3,3);INSERT INTO my_table VALUES (1,3); 4. INSERT INTO my_table VALUES (1,2); 5. INSERT INTO my_table VALUES (4,NULL); 6. INSERT INTO my_table VALUES (NULL,3); 7. INSERT INTO my_table VALUES (NULL,NULL);

Mark only one oval.

- command 4 will report an error.
- commands 3, 5 and 7 will report an error.
- commands 4 and 6 will report an error.
- commands 4, 5, 6 and 7 will report an error.

24. We create the tables "department" and "instructor" in phpmyadmin. To create the foreign key we go to the ...

Mark only one oval.

- ... "Relation View" menu.
- ... "Foreign Key" menu.
- ... "Referential Integrity" menu.
- ... "Table Connection" menu.

6 Unit 6: Intermediate SQL 2

Course: Database management
Unit: Intermediate SQL 2
Material: Referential integrity and data types

1. In a database, create the department and instructor tables connected with a foreign key.

```
create table department
 (dept_name  varchar(20),
  building  varchar(15),
  budget          numeric(12,2) check (budget > 0),
  primary key (dept_name)
 );

create table instructor
 (ID   varchar(5),
  name   varchar(20) not null,
  dept_name  varchar(20),
  salary   numeric(8,2) check (salary > 29000),
  primary key (ID),
  foreign key (dept_name) references department(dept_name)
 );

insert into department values ('Infor.', 'C', 30000);

insert into instructor values ('11111', 'Jaume', 'Infor.', 30000);
```

2. Is it possible to delete the row in the department table?
3. Is it possible to drop the department table?
4. Is it possible to update the name of the department?

```
update department set dept_name='Inform.' where dept_name='Infor.';
```

5. Modify the instructor table in such a way that when a row of the department table is deleted, the instructors of that department are also deleted. If you don't know how to modify the instructor table, you can drop it and create it again.
6. Modify the instructor table in such a way that when a row of the department table is deleted, the dept_name column of the corresponding instructors is set to null.
7. Modify the instructor table in such a way that when the name of a department is updated in the department table, it is also updated in the instructor table.
8. Drop the department and instructor tables and create them again with the code of exercise number 1. Now disable foreign key checks and attempt to delete the row of the department table. Does it work?

9. Create a table named “my_table” with a column named “my_date” (type:date), “my_time” (type:time) and “my_timestamp” (type timestamp).
10. Insert a row that contains a value for each of the three columns of the table.
11. Now insert a row that contains a value only for the first two columns. What is the default value for the timestamp column?
12. Explain how to extract the fields year, month, day, hour, minute and second from a timestamp value.
13. Explain the use of the following functions: current_date, current_time, localtime, current_timestamp, localtimestamp.
14. What is the use of the keyword “interval”? Give an example.
15. Drop the tables instructor and department and create them again as in exercise 1. Create an index named “instructor_name”. Why are indexes important?
16. Create a table called my_table with a column called my_column of type “text”. Introduce a text of several pages into the table. Then use select to show the text.
17. Load the university database and create a new table and a view with the following statements:

```
create table t1 as (select * from instructor where dept_name='Music');
create view v1 as (select * from instructor where dept_name='Music');
```

Show the contents of t1 and then show the contents of v1. Delete Mozart from the instructor table. Show the contents of t1 and v1 again and explain what happened.

Course: Database management
Unit: Intermediate SQL 2
Assignment: Mini-league
Teamwork: Groups of two or three

1. Create a database with name “MiniPremiereLeague” and use it in the next exercises.

2. Create the following table using MariaDB/MySQL and engine InnoDB:
TEAMS (tid, tname, tyear)
Explanation:

- It’s a table to store data about football teams.
- Fields:
  - tid: **Primary Key** of the table.
  - tname: Name of the football team. This field must be **NOT NULL** and **unique**.
  - tyear: Year of creation of the team. This year must be **higher than 1875**.

Very Important: Before MariaDB 10.2.1 constraint expressions were accepted in the syntax but ignored. https://mariadb.com/kb/en/library/constraint/

3. Create the following table using MariaDB/MySQL and engine InnoDB:
MATCHES (mid, mtidhome*, mtidaway*, mgoalshome, mgoalsaway, mdate)
Explanation:

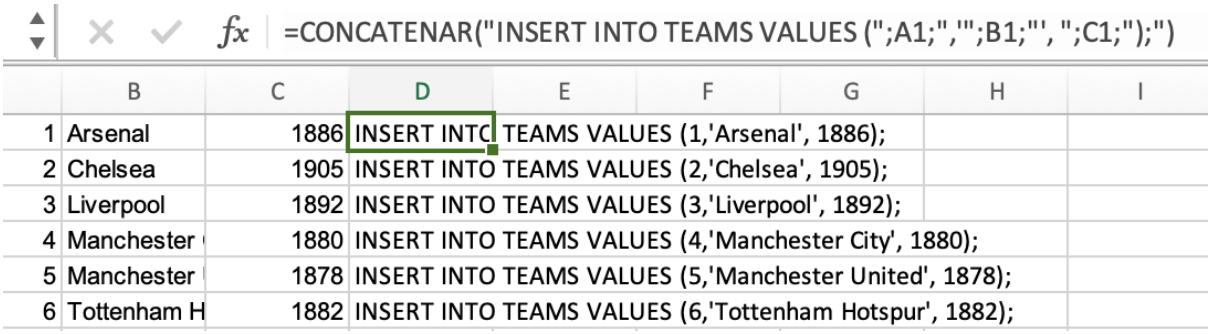
- It’s a table to store data about football teams.
- Fields:
  - mid: Primary Key of the table.
  - mtidhome: ID of the team playing at home. It’s a **Foreign Key**. If the registry with the **primary key** is **deleted** in the referenced table you muset set NULL values, but if it’s **updated** the foreign key must be updated on cascade.
  - mtidaway: ID of the team playing as visitor. It’s a **Foreign Key**. If the registry with the **primary key** is **deleted** in the referenced table you muset set NULL values, but if it’s **updated** the foreign key must be updated on cascade.
  - mgoalshome: Number of goals scored as local team. **Default value** must be **0**.
  - mgoalsaway:Number of goals scored as visitor. **Default value** must be **0**.
  - mdate: Date of the match. This field must be **NOT NULL**.

4. Insert the following data inside TEAMS:

1	Arsenal	1886
2	Chelsea	1905

3	Liverpool	1892
4	Manchester City	1880
5	Manchester United	1878
6	Tottenham Hotspur	1882

Clue: Make an Excel file to generate the queries:



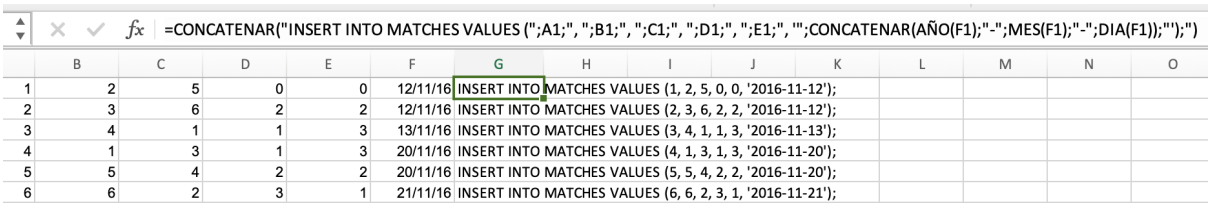
5. Insert the following data inside MATCHES:

1	2	5	0	0	12/11/2016
2	3	6	2	2	12/11/2016
3	4	1	1	3	13/11/2016
4	1	3	1	3	20/11/2016
5	5	4	2	2	20/11/2016
6	6	2	3	1	21/11/2016
7	2	1	0	1	27/11/2016
8	3	4	1	0	27/11/2016
9	6	5	2	2	28/11/2016
10	1	6	3	3	3/12/2016
11	3	5	2	2	3/12/2016
12	4	2	1	0	4/12/2016
13	2	3	1	0	10/12/2016
14	5	1	0	1	10/12/2016
15	6	4	1	1	11/12/2016
16	5	2	1	2	7/1/2017
17	6	3	2	1	7/1/2017

18	1	4	1	2	8/1/2017
19	3	1	0	1	14/1/2017
20	4	5	1	0	14/1/2017
21	2	6	0	0	15/1/2017
22	1	2	0	0	21/1/2017
23	4	3	5	1	21/1/2017
24	5	6	1	2	22/1/2017
25	6	1	2	1	4/2/2017
26	5	3	3	1	4/2/2017
27	2	4	3	1	5/2/2017
28	3	2	2	2	11/2/2017
29	1	5	1	0	11/2/2017
30	4	6	1	1	12/2/2017
31	5	2	0	1	18/11/2017
32	4	1	0	0	18/11/2017
33	3	6	0	0	19/11/2017
34	6	4	6	0	25/11/2017
35	2	3	3	1	25/11/2017
36	1	5	1	1	26/11/2017
37	5	6	1	2	2/12/2017
38	4	3	3	0	2/12/2017
39	1	2	2	0	3/12/2017
40	6	1	1	0	9/12/2017
41	4	2	1	1	9/12/2017
42	3	5	3	1	10/12/2017
43	5	4	2	1	16/12/2017
44	2	6	1	1	16/12/2017
45	1	3	0	0	17/1/2018
46	2	5	2	0	13/1/2018

47	1	4	1	1	14/1/2018
48	6	3	1	1	14/1/2018
49	4	6	2	1	20/1/2018
50	3	2	3	0	21/1/2018
51	5	1	1	0	21/1/2018
52	6	5	1	0	27/1/2018
53	3	4	3	3	28/1/2018
54	2	1	1	0	28/1/2018
55	1	6	0	2	3/2/2018
56	2	4	1	0	4/2/2018
57	5	3	2	1	4/2/2018
58	4	5	3	2	10/2/2018
59	6	2	2	3	11/2/2018
60	3	1	1	0	11/2/2018

Clue: Make an Excel file to generate the queries:



6. Select the number of matches per team at home and away. Clue: Divide and conquer.

```
+-------------------+---------------+-----------------+
| tname             | NMatchesLocal | NMatchesVisitor |
+-------------------+---------------+-----------------+
| Arsenal           |            10 |              10 |
| Chelsea           |            10 |              10 |
| Liverpool         |            10 |              10 |
| Manchester City   |            10 |              10 |
| Manchester United |            10 |              10 |
| Tottenham Hotspur |            10 |              10 |
+-------------------+---------------+-----------------+
6 rows in set (0.004 sec)
```

7. DANGER (for PROS): Show me the classification of the league for the season 2016/2017.

TeamName	Points	Won	Drawn	Lost	GF	GA	GD
Tottenham Hotspur	28	7	7	0	27	14	13
Arsenal	22	6	4	4	16	13	3
Chelsea	20	5	5	4	14	13	1
Manchester City	20	5	5	4	19	20	-1
Liverpool	13	3	4	7	17	26	-9
Manchester United	8	1	5	8	14	21	-7

6 rows in set (0.004 sec)

Points: Won*3 + Drawn*1
GF: Goals For, sometimes used in place of GS (Goals Scored).
GA – Goals Against (i.e., number of goals conceded by a team).
GD – Goal Difference (i.e., difference between GF and GA, and sometimes denoted by +/-).
Clue: You can use CASE or IF.

8. DANGER (for PROS): Show me the classification of the league for the season 2017/2018.

TeamName	Points	Won	Drawn	Lost	GF	GA	GD
Chelsea	20	6	2	2	13	10	3
Tottenham Hotspur	18	5	3	2	17	8	9
Liverpool	13	3	4	3	13	13	0
Manchester City	13	3	4	3	14	17	-3
Manchester United	10	3	1	6	10	15	-5
Arsenal	7	1	4	5	4	8	-4

6 rows in set (0.004 sec)

Points: Won*3 + Drawn*1
GF: Goals For, sometimes used in place of GS (Goals Scored).
GA – Goals Against (i.e., number of goals conceded by a team).
GD – Goal Difference (i.e., difference between GF and GA, and sometimes denoted by +/-).
Clue: You can use CASE or IF.

9. What does it happen if you delete the team with id 1 in the table TEAMS?

10. Alter the relation “MATCHES” to avoid deleting teams if they are in use in the table matches (drop them and insert them again with the new values).

11. Try to delete now the team with id 1 in the table TEAMS and see what happens.

12. Can you disable (and enable) foreign key checks temporary? (Surf the web to answer this question).

13. Changes the primary key of the teams to:

- ‘ARS’ = Arsenal
- ‘CHE’ = Chelsea
- ‘LIV’ = Liverpool

- ‘MCI’ = Manchester City
- ‘MUN’ = Manchester United
- ‘TOT’ = Tottenham Hotspur

What you should do with the foreign keys?

Course: Database management
Unit: Intermediate SQL 2
Material: Index importance

Load the TENNIS.sql and TENNIS_NOPK_NOFK.sql . They are the same database but the second one has no primary keys and no foreign keys. As a consequence, the second has no indexes. Run the query you will find below in both databases and note how long it takes to produce a result.

```
SELECT CONCAT_WS(' ', P1.p_name, P1.p_surname) AS Player1,
CONCAT_WS(' ', P2.p_name, P2.p_surname) AS Player2, t_name AS Tournament, t_end_date
AS `End date`, IF (mr_winner = R1.r_num, CONCAT_WS(' ', 'Winner:', P1.p_name,
P1.p_surname), CONCAT_WS(' ', 'Winner:', P2.p_name, P2.p_surname)) AS Winner
FROM REGISTRATIONS R1, REGISTRATIONS R2, MATCHES, MATCH_RESULTS,
TOURNAMENTS, PLAYERS P1, PLAYERS P2
WHERE
m_id = mr_m_id AND
  ((R1.r_num = m_r_num1 AND R2.r_num = m_r_num2) OR
    (R1.r_num = m_r_num2 AND R2.r_num = m_r_num1) ) AND
       t_id = m_t_id AND
       t_num_rounds = m_round AND
       t_type = 'Singles' AND
       R2.r_p_id = P2.p_id AND
       R1.r_p_id = P1.p_id AND
       P1.p_id < P2.p_id;
```

Course: Database management
Unit: Intermediate SQL 2
Material: Authorization

1. Connect to MariaDB as root.
2. Create a user named “my_user” identified by “my_password”.
3. Create a database named “my_database”.
4. Open a second shell. Connect to MariaDB as “my_user”.
5. Try to use the database my_database. Does it work?
6. Go back to the terminal with the root user and grant all privileges to “my_user” to use “my_database”. Grant access from any host (‘my_user’@’%’) .
7. Go to the ‘my_user’ terminal and connect to the database.
8. With the root terminal user, create a table named “my_table” in “my_database”. create table my_table(c1 int);
9. In the my_user terminal, see if you can run a select query on ‘my_table’ with ‘my_user’.
10. With the user root, revoke the select privilege on my_database to user my_user. revoke select on my_database.* from my_user;
11. In the ‘my_user’ terminal, disconnect from MariaDB and then connect again. Use my_database. Try to repeat the select query on my_table. What happens?
12. Can you run an insert? Why? insert into my_table values (1);
13. Find a list of the privileges that can be granted and revoked.
14. In the my_user terminal, run the “show grants” command. What can you see?
15. In the root terminal, run the “show grants” command. What can you see?
16. What is the meaning of the “with grant” keywords?
17. In the root terminal, create a new user my_user_2 identified by my_password_2.
18. In the my_user terminal, try to grant the insert privilege on my_database to my_user_2. Does it work? Why?

Course: Database management
Unit: Intermediate SQL 2
Material: Roles

You can find many of the commands for this exercise in the following link
https://severalnines.com/database-blog/database-user-management-managing-roles-mariadb

1. Run a MariaDB server.
2. From a terminal, connect to MariaDB as root.
3. Run: SHOW GRANTS; to see the grants for the current user.
4. Run: SELECT CURRENT_ROLE; to see the current role.
5. Run: SELECT user FROM mysql.user WHERE is_role='Y'; to see the list of roles.
6. Run: SELECT * FROM information_schema.applicable_roles; to see the list of available roles.
7. Run: SELECT * FROM information_schema.enabled_roles; to see the list of active roles.
8. Run: SELECT * FROM mysql.roles_mapping; to see the roles that have been granted to the users.
9. Create a database named testing;
10. Create a user ‘testuser’@’%’ identified by ‘password’.
11. Check the grants for the new user. SHOW GRANTS for ‘testuser’@'%';
12. Open a second terminal.
13. In the new terminal, connect to MariaDB with the new user testuser.
14. In the testuser terminal, try to use the testing database. Does it work?
15. Move back to the root terminal. Create a role named “qateam”.
16. Give the select, insert, update and delete privileges on the testing database to the role qateam.
17. Move back to the testuser terminal. Try to change the role to qateam. Does it work?
18. Move back to the root terminal and grant the qateam role to testuser.
19. Move back the testuser terminal and try to change the role to qateam. Does it work this time?
20. Try to use the testing database. Does it work?
21. Check the grants for the current user.
22. Check the current role.
23. Move back to the root terminal.
24. Check the grants for the current user;
25. Check the current role.
26. Change the role to qateam.
27. Check the grants for the current user.
28. Check the current role.
29. Show the list of roles.
30. Show the list of available roles.

31. Show the list of active roles.
32. Show the roles that have been granted to the users.

Course: Database management
Unit: Intermediate SQL 2
Material: Dates

In order to have the same database, please, run the following SQL code:

```
DROP DATABASE IF EXISTS `SQL1NORMALDATES`;
CREATE DATABASE `SQL1NORMALDATES`;
USE `SQL1NORMALDATES`;

CREATE TABLE `DEPARTMENTS` (
  `num` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `town_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `DEPARTMENTS` (`num`, `name`, `town_code`) VALUES
(10, 'ACCOUNTING', 'SVQ'),
(20, 'RESEARCH', 'MAD'),
(30, 'SALES', 'BCN'),
(40, 'PRODUCTION', 'BIO');

CREATE TABLE `EMPLOYEES` (
  `num` int(11) NOT NULL,
  `surname` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `manager` int(11) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `salary` int(11) DEFAULT NULL,
  `commission` int(11) DEFAULT NULL,
  `dept_num` int(11) DEFAULT NULL,
  `occu_code` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `EMPLOYEES` (`num`, `surname`, `name`, `manager`, `start_date`, `salary`,
`commission`, `dept_num`, `occu_code`) VALUES
(1000, 'PITT', 'BRAD', NULL, '2004-01-01', 1040, NULL, 20, NULL),
(7369, 'SÁNCHEZ', 'SERGIO', 8001, '2010-12-17', 1040, NULL, 20, 'EMP'),
(7499, 'ARROYO', 'MARTA', 7698, '2010-02-20', 1500, 390, 30, 'SAL'),
(7521, 'GONZÁLEZ', 'RAUL', 7782, '2011-02-22', 1625, 650, 30, 'SAL'),
(7566, 'JIMÉNEZ', 'JUAN JOSÉ', 1000, '2017-04-02', 2900, NULL, 20, 'MAN'),
(7654, 'MARTÍN', 'MONICA', 7698, '2017-09-29', 1600, 1020, 30, 'SAL'),
(7698, 'GUASP', 'BARTOLOME', 1000, '2017-05-01', 3005, NULL, 30, 'MAN'),
(7782, 'CEREZO', 'JOSÉ', 1000, '2010-06-09', 2885, NULL, 10, 'MAN'),
(7788, 'GIL', 'JAVIER', 8000, '2010-11-09', 3000, NULL, 20, NULL),
(7844, 'TOVAR', 'LUIS', 7698, '2018-09-08', 1350, 0, 30, 'SAL'),
(7876, 'ALONSO', 'FERNANDO', 7788, '2018-09-23', 1430, NULL, 20, 'EMP'),
(7900, 'JIMENO', 'XAVIER', 8001, '2017-12-03', 1335, NULL, 30, 'EMP'),
(7902, 'FERNÁNDEZ', 'ANA', 8000, '2016-12-03', 3000, NULL, 20, NULL),
(7934, 'MUÑOZ', 'ANTONIA', 8001, '2016-01-23', 1690, NULL, 10, 'EMP'),
(8000, 'BANDERAS', 'ANTONIO', 1000, '2017-01-09', 2885, NULL, 20, 'MAN'),
(8001, 'RUIZ', 'FERNANDA', 1000, '2018-06-10', 2885, NULL, 20, 'MAN');

CREATE TABLE `OCCUPATIONS` (
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `OCCUPATIONS` (`code`, `name`) VALUES
('ANA', 'ANALIST'),
('EMP', 'EMPLOYEE'),
('MAN', 'MANAGER'),
('PRE', 'PRESIDENT'),
('SAL', 'SALESMAN');

CREATE TABLE `TOWNS` (
```

```
  `code` varchar(3) NOT NULL,
  `name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `TOWNS` (`code`, `name`) VALUES
('BCN', 'BARCELONA'),
('BIO', 'BILBAO'),
('MAD', 'MADRID'),
('SVQ', 'SEVILLA');

ALTER TABLE `DEPARTMENTS`
  ADD PRIMARY KEY (`num`),
  ADD KEY `town_code` (`town_code`);

ALTER TABLE `EMPLOYEES`
  ADD PRIMARY KEY (`num`),
  ADD KEY `dept_num` (`dept_num`),
  ADD KEY `manager` (`manager`),
  ADD KEY `occu_code` (`occu_code`);

ALTER TABLE `OCCUPATIONS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `TOWNS`
  ADD PRIMARY KEY (`code`);

ALTER TABLE `DEPARTMENTS`
  ADD CONSTRAINT `DEPARTMENTS_ibfk_1` FOREIGN KEY (`town_code`) REFERENCES `TOWNS`
(`code`);

ALTER TABLE `EMPLOYEES`
  ADD CONSTRAINT `EMPLOYEES_ibfk_1` FOREIGN KEY (`dept_num`) REFERENCES `DEPARTMENTS`
(`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_2` FOREIGN KEY (`manager`) REFERENCES `EMPLOYEES` (`num`),
  ADD CONSTRAINT `EMPLOYEES_ibfk_3` FOREIGN KEY (`occu_code`) REFERENCES `OCCUPATIONS`
(`code`);
```

Note that the results depend on the date that you run them…

0. To do this practice we need to activate root access and to load the time zone table into MySQL/MariaDB:

Activate root access:

- $> su
- root> mysql
- mysql> FLUSH PRIVILEGES;
- mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('alualualu');
- mysql> exit

Load the time zone table:

- root> mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql

Screenshot:

```
root@alumne-VirtualBox:/home/alumne# mysql -h localhost -u root -p
Enter password:
alumne@alumne-VirtualBox:~$ su
Password:
root@alumne-VirtualBox:/home/alumne# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.25-0ubuntu0.18.10.2 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('alualualu');
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> exit
Bye
root@alumne-VirtualBox:/home/alumne# service mysql restart
root@alumne-VirtualBox:/home/alumne# mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql
Enter password:
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
root@alumne-VirtualBox:/home/alumne#
```

More references here: https://mariadb.com/kb/en/library/mysql_tzinfo_to_sql/

1. Select the last day of the present month.

```
+------------+
| MyDay      |
+------------+
| 2019-01-31 |
+------------+
```

2. Select the last day of the month three months before today.

```
+------------+
| MyDay      |
+------------+
| 2018-10-31 |
+------------+
```

3. Show the date of exercise 2 with format “Name_of_month day, year with 2 digits”.

```
+----------------+
| MyDay          |
+----------------+
| October 31, 18 |
+----------------+
```

4. Write a query to convert 680001 days in a date.

```
+------------+
| MyDay      |
+------------+
| 1861-10-12 |
+------------+
```

6. Use CONVERT_TZ to convert the current date/time (UTC) to Panama. Clue: Visit this link.

```
+---------------------+---------------------+
| DateMadrid          | DatePanama          |
+---------------------+---------------------+
| 2019-01-31 13:19:58 | 2019-01-31 07:19:58 |
+---------------------+---------------------+
```

7. Use CONVERT_TZ to convert the current date/time (UTC) to Sydney. Clue: Visit this link.

```
+---------------------+---------------------+
| DateMadrid          | DateSydney          |
+---------------------+---------------------+
| 2019-01-31 13:18:29 | 2019-01-31 23:18:29 |
+---------------------+---------------------+
```

8. Subtract 3 hours 25 minutes to the current date/time using DATE_SUB.

```
+---------------------+---------------------+
| Now                 | MyTimestamp         |
+---------------------+---------------------+
| 2019-01-31 13:25:57 | 2019-01-31 03:00:57 |
+---------------------+---------------------+
```

9. Which day of the year (1, 2, 3, etc., 365) is today (example for 31-01-2019 not for today)?

```
+---------------------+----------------+
| Today               | TodatDayNumber |
+---------------------+----------------+
| 2019-01-31 13:27:45 |             31 |
+---------------------+----------------+
```

10. Can you convert a String to a Date and/or Time? Write an example.

11. Search on the Internet what's a period and make an example using the function PERIOD_ADD.

12. Difference in days between the employee who started in the first place and the employee who started in the last place.

```
+-------+
| MyDay |
+-------+
|  5379 |
+-------+
```

13. Select the dates inside the field EMPLOYEES.start_date that are Tuesday.

```
+-----------+--------+------------+
| surname   | name   | start_date |
+-----------+--------+------------+
| GONZÁLEZ  | RAUL   | 2011-02-22 |
| GIL       | JAVIER | 2010-11-09 |
+-----------+--------+------------+
```

14. Select the data of the older employee in the enterprise.

num	surname	name	manager	start_date	salary	commission	dept_num	occu_code
1000	PITT	BRAD	NULL	2004-01-01	1040	NULL	20	NULL

15. Select the employees' name, surname and years working in our enterprise (order by those years descendent).

Option 1:

name	surname	Years
BRAD	PITT	15.9479
MARTA	ARROYO	9.8055
JOSÉ	CEREZO	9.5068
JAVIER	GIL	9.0877
SERGIO	SÁNCHEZ	8.9836
RAUL	GONZÁLEZ	8.8000
ANTONIA	MUÑOZ	3.8795
ANA	FERNÁNDEZ	3.0164
ANTONIO	BANDERAS	2.9151
JUAN JOSÉ	JIMÉNEZ	2.6877
BARTOLOME	GUASP	2.6082
MONICA	MARTÍN	2.1945
XAVIER	JIMENO	2.0164
FERNANDA	RUIZ	1.4986
LUIS	TOVAR	1.2521
FERNANDO	ALONSO	1.2110

Option 2:

name	surname	Years
BRAD	PITT	15 years 12 months 09 days
MARTA	ARROYO	09 years 10 months 19 days
JOSÉ	CEREZO	09 years 07 months 02 days
JAVIER	GIL	09 years 01 months 30 days
SERGIO	SÁNCHEZ	08 years 12 months 23 days
RAUL	GONZÁLEZ	08 years 10 months 17 days
ANTONIA	MUÑOZ	03 years 11 months 17 days
ANA	FERNÁNDEZ	03 years 01 months 06 days
ANTONIO	BANDERAS	02 years 11 months 30 days
JUAN JOSÉ	JIMÉNEZ	02 years 09 months 08 days
BARTOLOME	GUASP	02 years 08 months 10 days
MONICA	MARTÍN	02 years 03 months 12 days
XAVIER	JIMENO	02 years 01 months 06 days
FERNANDA	RUIZ	01 years 07 months 01 days
LUIS	TOVAR	01 years 04 months 02 days
FERNANDO	ALONSO	01 years 03 months 18 days

16. Write a query to show EMPLOYEES.start_date in three columns: year, month and day.

Year	Month	Day
2004	1	1
2010	12	17
2010	2	20
2011	2	22
2017	4	2
2017	9	29
2017	5	1
2010	6	9
2010	11	9
2018	9	8
2018	9	23
2017	12	3
2016	12	3
2016	1	23
2017	1	9
2018	6	10

17. Write a query to show the employees that joined the enterprise in June.

surname	name	start_date
CEREZO	JOSÉ	2010-06-09
RUIZ	FERNANDA	2018-06-10

18. Write the date of exercise 17 in the next format:

surname	name	start_date
CEREZO	JOSÉ	Wednesday 9th June 2010 00:00:00
RUIZ	FERNANDA	Sunday 10th June 2018 00:00:00

19. Write a query to get the year and number of employees who began working that year.

year	num
2004	1
2010	4
2011	1
2016	2
2017	5
2018	3

20. Write a query to get the maximum number of employees who started working in our enterprise in a year.

max_num
5

21. Write a query to get the year in which more employees joined our enterprise.

year
2017

22. Show employees who are manager of other employees and the time in years that they are working in the enterprise.

num	name	surname	Years
1000	BRAD	PITT	15.9479
7698	BARTOLOME	GUASP	2.6082
7782	JOSÉ	CEREZO	9.5068
7788	JAVIER	GIL	9.0877
8000	ANTONIO	BANDERAS	2.9151
8001	FERNANDA	RUIZ	1.4986

23. Show employees who are manager of other employees working in the enterprise for more than 5 years.

num	name	surname
1000	BRAD	PITT
7782	JOSÉ	CEREZO
7788	JAVIER	GIL

24. Can you use BETWEEN keyword with dates. Write an example.

Course: Database management
Unit: Intermediate SQL 2
Test

Connect to mariadb as root user

Load the exam.sql script

1. (0.5) Add primary keys and foreign keys to the tables. It must not be possible to delete the teams that are referenced in the MATCHES table. It must be possible to update the id's of the teams that are referenced in the matches table.

```
mysql> alter table TEAMS add primary key (tid);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table MATCHES add primary key (mid);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table MATCHES add foreign key (mtidhome) references TEAMS (tid) on delete
restrict on update set null;
Query OK, 60 rows affected (0.05 sec)
Records: 60  Duplicates: 0  Warnings: 0

mysql> alter table MATCHES add foreign key (mtidaway) references TEAMS (tid) on delete
restrict on update set null;
Query OK, 60 rows affected (0.05 sec)
Records: 60  Duplicates: 0  Warnings: 0
```

2. (0.5) Create a user with your name identified by a password of your choice.

```
mysql> create user 'jbarcelo'@'%' identified by 'secret';
Query OK, 0 rows affected (0.00 sec)
```

3. (0.5) Create a role named student

```
mysql> create role student;
Query OK, 0 rows affected (0.00 sec)
```

4. (0.5) Grant select, update permissions on all the tables of the database to the role you created.

```
mysql> grant select, update on MiniPremierLeague.* to 'student';
```

Query OK, 0 rows affected (0.00 sec)

5. (0.5) Grant the student role to your user

mysql> grant student to jbarcelo;
Query OK, 0 rows affected (0.00 sec)

6. (0.5) Connect to the database using your user. Activate the student role.

jbarcelo@cloudshell:~$ mysql -ujbarcelo -h 172.18.0.2 -p
Enter password:

mysql> set role student;
Query OK, 0 rows affected (0.00 sec)

7. (3) For each team, compute the number of matches that the team has scored 0 goals in the season 16-17. Order the results from the smaller to the larger.

mysql> select tid, tname, (select count(*) from MATCHES where mtidhome=tid and mgoalshome=0 and mid between 1 and 30) + (select count(*) from MATCHES where mtidaway=tid and mgoalsaway=0 and mid between 1 and 30) as zeros from TEAMS order by zeros;

tid	tname	zeros
1	Arsenal	1
4	Manchester City	1
6	Tottenham Hotspur	1
3	Liverpool	2
5	Manchester United	4
2	Chelsea	5

6 rows in set (0.00 sec)

8. (1) Design a query that computes the time difference in years, months, and days between the first match and the last match of the database.
The query should work even if the dates of the matches change.

mysql> select date_format(from_days(to_days(max(mdate))-to_days(min(mdate))),'%y years, %m months and %d days') as time from MATCHES;

time
01 years, 04 months and 01 days

```
+---------------------------------+
1 row in set (0.00 sec)
```

9. (1) Design a query that adds one month and ten days to the dates of all the matches of the database.

```
mysql> update MATCHES set mdate=mdate+interval 1 month + interval 10 day;
Query OK, 60 rows affected (0.00 sec)
Rows matched: 60  Changed: 60  Warnings: 0
```

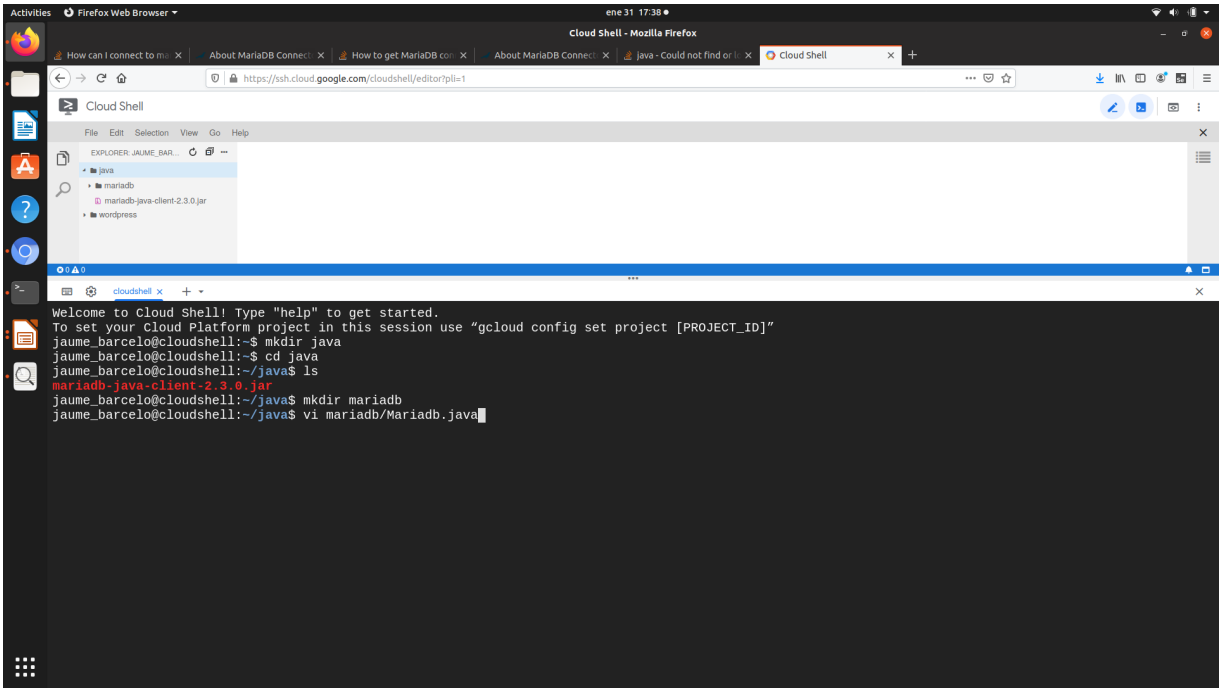
10. (1)Design a query to return the number of days in which more than one match was played.

```
mysql> with mdate_no_matches as (select mdate, count(*) as no_matches from MATC
HES group by mdate having no_matches>1) select count(*) days_with_more_than_1_m
atch from mdate_no_matches;
+----------------------------+
| days_with_more_than_1_match |
+----------------------------+
|                         20 |
+----------------------------+
1 row in set (0.00 sec)
```

7 Unit 7: Advanced SQL 1

Course: Database management
Unit: Advanced SQL 1
Material: SQL statements in a java application

The following steps show you how to create a java program that connects to MariaDB.



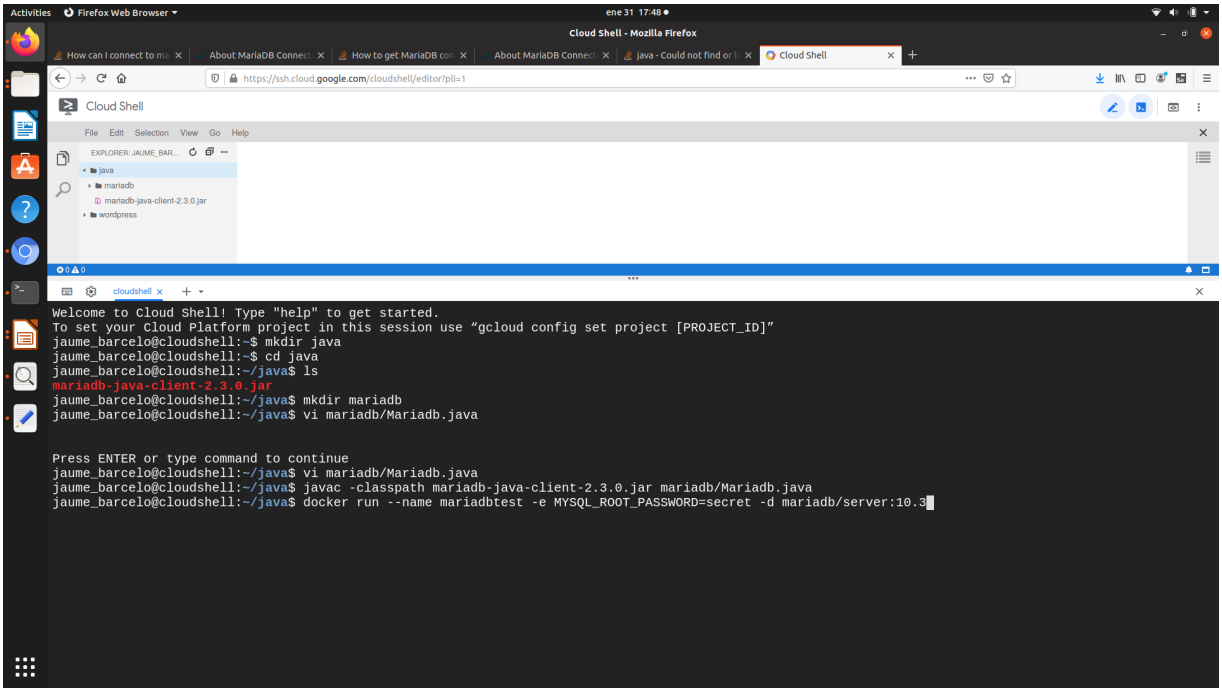
You can obtain the jar file running the following command
wget
https://downloads.mariadb.com/Connectors/java/connector-java-2.3.0/mariadb-java-client-2.3.0.jar

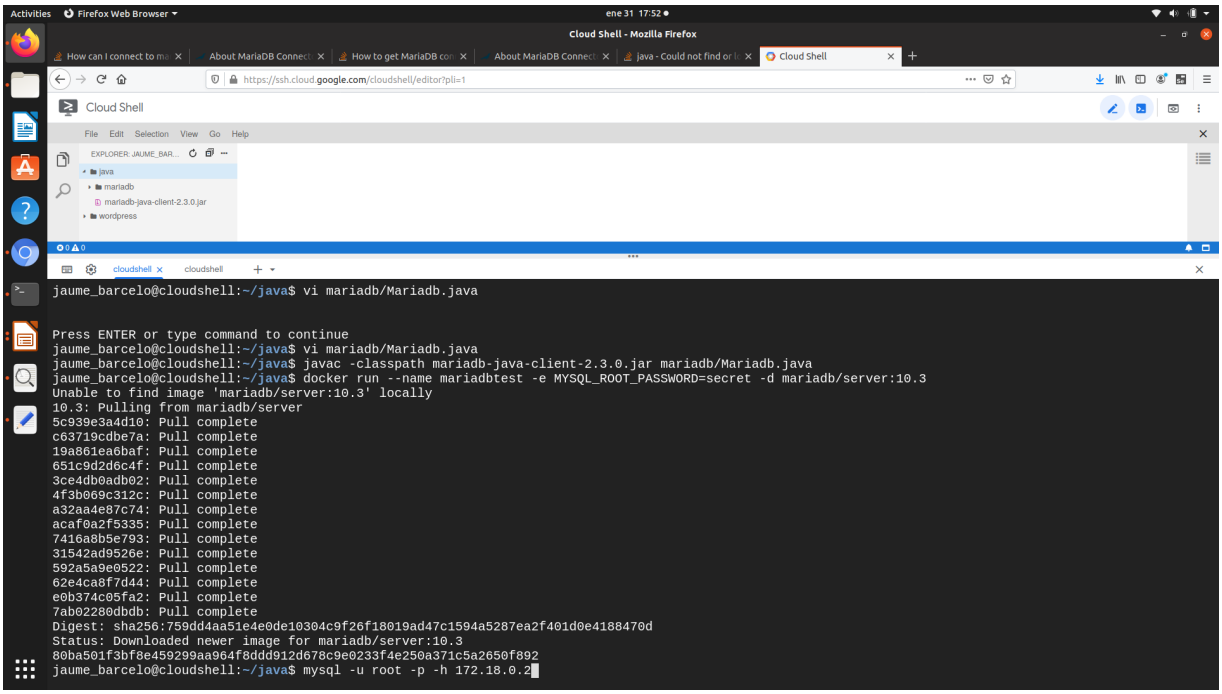
The following is example code to connect to the database

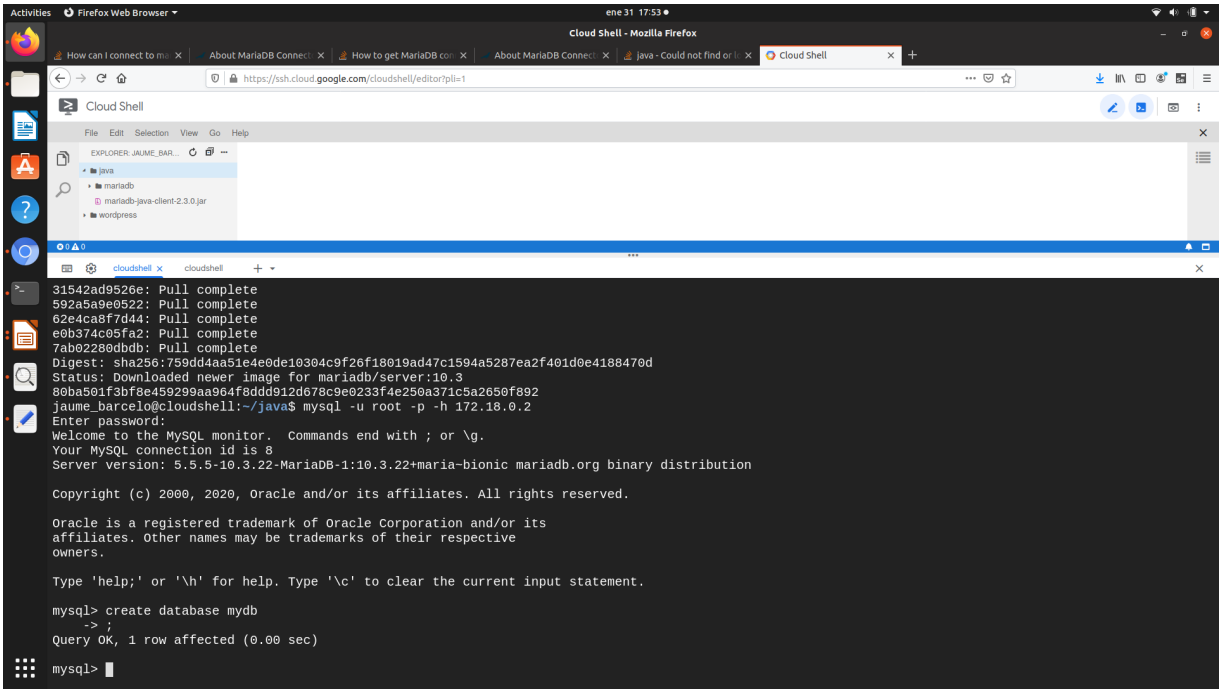
```
//STEP 1. Import required packages
package mariadb;

import java.sql.*;

public class Mariadb {
   // JDBC driver name and database URL

   static final String JDBC_DRIVER = "org.mariadb.jdbc.Driver";
   static final String DB_URL = "jdbc:mariadb://172.18.0.2/mydb";

   //  Database credentials
   static final String USER = "root";
   static final String PASS = "root";
```

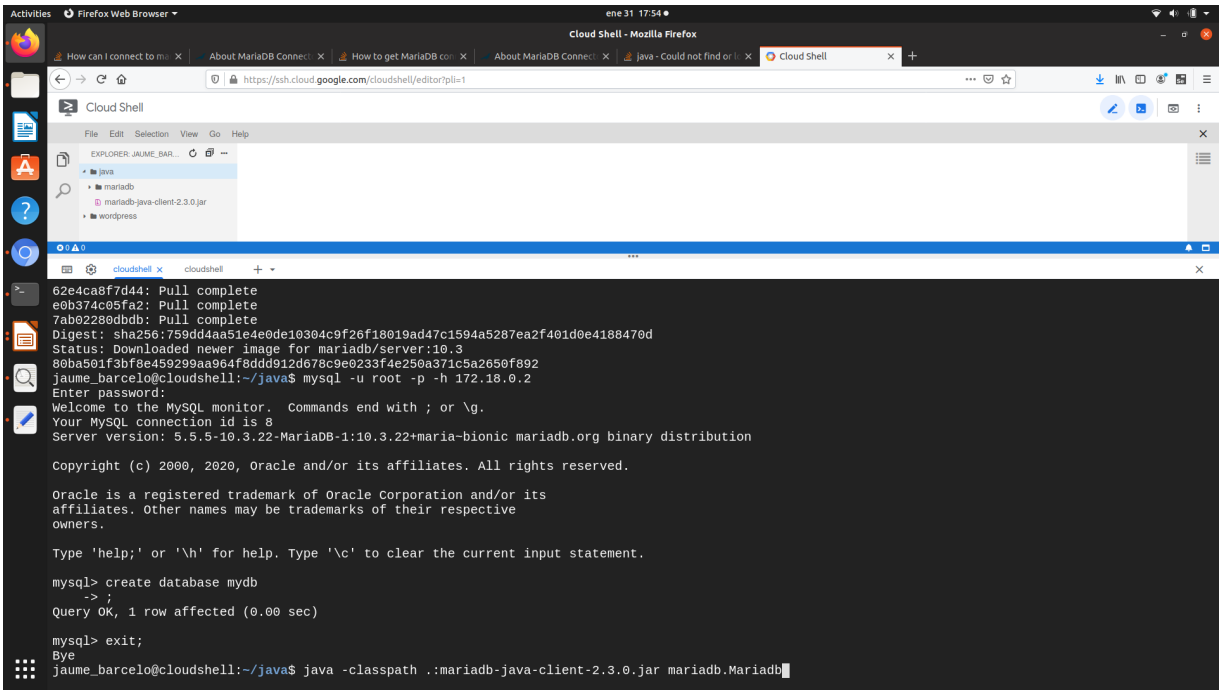
```
public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try {
      //STEP 2: Register JDBC driver
      Class.forName("org.mariadb.jdbc.Driver");

      //STEP 3: Open a connection
      System.out.println("Connecting to a selected database...");
      conn = DriverManager.getConnection(
            "jdbc:mariadb://172.18.0.2/mydb", "root", "secret");
      System.out.println("Connected database successfully...");

      //STEP 4: Execute a query
      System.out.println("Creating table in given database...");
      stmt = conn.createStatement();

      String sql = "CREATE TABLE REGISTRATION "
            + "(id INTEGER not NULL, "
            + " first VARCHAR(255), "
            + " last VARCHAR(255), "
            + " age INTEGER, "
            + " PRIMARY KEY ( id ))";

      stmt.executeUpdate(sql);
      System.out.println("Created table in given database...");
   } catch (SQLException se) {
      //Handle errors for JDBC
      se.printStackTrace();
   } catch (Exception e) {
      //Handle errors for Class.forName
      e.printStackTrace();
   } finally {
      //finally block used to close resources
      try {
         if (stmt != null) {
            conn.close();
         }
      } catch (SQLException se) {
      }// do nothing
      try {
         if (conn != null) {
```

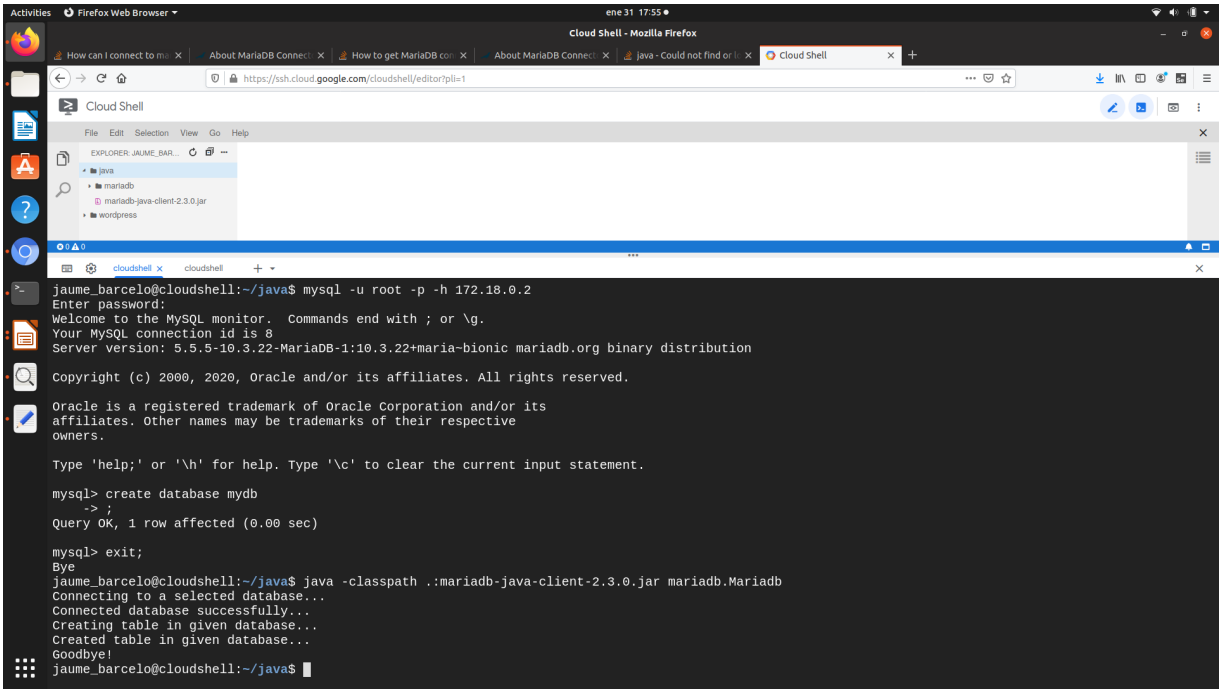
```
                conn.close();
            }
        } catch (SQLException se) {
            se.printStackTrace();
        }//end finally try
    }//end try
    System.out.println("Goodbye!");
  }//end main
}//end JDBCExample
```




Activities
Firefox Web Browser
ene 31 17:52
Cloud Shell - Mozilla Firefox
Cloud Shell
https://ssh.cloud.google.com/cloudshell/editor?pli=1
File Edit Selection View Go Help
EXPLORER: JAUME_BAR...
java
mariadb
mariadb-java-client-2.3.0.jar
wordpress
cloudshell
jaume_barcelo@cloudshell:~/java$ vi mariadb/Mariadb.java
Press ENTER or type command to continue
jaume_barcelo@cloudshell:~/java$ vi mariadb/Mariadb.java
jaume_barcelo@cloudshell:~/java$ javac -classpath mariadb-java-client-2.3.0.jar mariadb/Mariadb.java
jaume_barcelo@cloudshell:~/java$ docker run --name mariadbtest -e MYSQL_ROOT_PASSWORD=secret -d mariadb/server:10.3
Unable to find image 'mariadb/server:10.3' locally
10.3: Pulling from mariadb/server
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
3ce4db0adb02: Pull complete
4f3b069c312c: Pull complete
a32aa4e87c74: Pull complete
acaf0a2f5335: Pull complete
7416a8b5e793: Pull complete
31542ad9526e: Pull complete
592a5a9e0522: Pull complete
62e4ca8f7d44: Pull complete
e0b374c05fa2: Pull complete
7ab02280dbdb: Pull complete
Digest: sha256:759dd4aa51e4e0de10304c9f26f18019ad47c1594a5287ea2f401d0e4188470d
Status: Downloaded newer image for mariadb/server:10.3
80ba501f3bf8e459299aa964f8ddd912d678c9e0233f4e250a371c5a2650f892
jaume_barcelo@cloudshell:~/java$ mysql -u root -p -h 172.18.0.2


Activities
Firefox Web Browser
ene 31 17:53
Cloud Shell - Mozilla Firefox
Cloud Shell
https://ssh.cloud.google.com/cloudshell/editor?pli=1
File Edit Selection View Go Help
EXPLORER: JAUME_BAR...
java
mariadb
mariadb-java-client-2.3.0.jar
wordpress
cloudshell
31542ad9526e: Pull complete
592a5a9e0522: Pull complete
62e4ca8f7d44: Pull complete
e0b374c05fa2: Pull complete
7ab02280dbdb: Pull complete
Digest: sha256:759dd4aa51e4e0de10304c9f26f18019ad47c1594a5287ea2f401d0e4188470d
Status: Downloaded newer image for mariadb/server:10.3
80ba501f3bf8e459299aa964f8ddd912d678c9e0233f4e250a371c5a2650f892
jaume_barcelo@cloudshell:~/java$ mysql -u root -p -h 172.18.0.2
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.5.5-10.3.22-MariaDB-1:10.3.22+maria~bionic mariadb.org binary distribution
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database mydb
-> ;
Query OK, 1 row affected (0.00 sec)
mysql>


Activities
Firefox Web Browser
ene 31 17:54
Cloud Shell - Mozilla Firefox
Cloud Shell
https://ssh.cloud.google.com/cloudshell/editor?pli=1
62e4ca8f7d44: Pull complete
e0b374c05fa2: Pull complete
7ab02280dbdb: Pull complete
Digest: sha256:759dd4aa51e4e0de10304c9f26f18019ad47c1594a5287ea2f401d0e4188470d
Status: Downloaded newer image for mariadb/server:10.3
80ba501f3bf8e459299aa964f8ddd912d678c9e0233f4e250a371c5a2650f892
jaume_barcelo@cloudshell:~/java$ mysql -u root -p -h 172.18.0.2
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.5.5-10.3.22-MariaDB-1:10.3.22+maria~bionic mariadb.org binary distribution
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database mydb
-> ;
Query OK, 1 row affected (0.00 sec)
mysql> exit;
Bye
jaume_barcelo@cloudshell:~/java$ java -classpath .:mariadb-java-client-2.3.0.jar mariadb.Mariadb


Activities
Firefox Web Browser
ene 31 17:55
Cloud Shell - Mozilla Firefox
Cloud Shell
https://ssh.cloud.google.com/cloudshell/editor?pli=1
jaume_barcelo@cloudshell:~/java$ mysql -u root -p -h 172.18.0.2
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.5.5-10.3.22-MariaDB-1:10.3.22+maria~bionic mariadb.org binary distribution
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database mydb
-> ;
Query OK, 1 row affected (0.00 sec)
mysql> exit;
Bye
jaume_barcelo@cloudshell:~/java$ java -classpath .:mariadb-java-client-2.3.0.jar mariadb.Mariadb
Connecting to a selected database...
Connected database successfully...
Creating table in given database...
Created table in given database...
Goodbye!
jaume_barcelo@cloudshell:~/java$

Course: Database management
Unit: Advanced SQL 1
Assignment: JDBC project
Teamwork: Groups of two or three

In groups of two, develop a project that involves the connection of a programming language with a database. You can use the example of the connection from java to mariadb as a starting point. Prepare a report and a 10 minutes presentation to explain your work to your classmates.

Course: Database management
Unit: Advanced SQL 1
Assignment: JDBC project video explanation
Teamwork: Groups of two or three

Prepare a video explanation and demonstration of your project.

8 Unit 8: Advanced SQL 2

Course: Database management
Unit: Advanced SQL 2
Material: PostgreSQL
Teamwork: Groups of two or three

Follow these instructions to create a database in postgres, load the the example “dvdrental” database and connect to the database

```
docker run --rm   --name pg-docker -e POSTGRES_PASSWORD=docker -d -p 5432:5432 postgres

mkdir dvdrental

cd dvdrental/

wget https://sp.postgresqltutorial.com/wp-content/uploads/2019/05/dvdrental.zip

unzip dvdrental.zip

docker cp dvdrental/dvdrental.tar  pg-docker:/tmp/

docker exec -it pg-docker createdb -U postgres dvdrental

docker exec -it pg-docker pg_restore -U postgres -d dvdrental /tmp/dvdrental.tar

docker exec -it pg-docker psql -h localhost -U postgres -d postgres

\c dvdrental

\dt
```

Course: Database management
Unit: Advanced SQL 2
Material: PostgreSQL PL/pgSQL
Teamwork: Groups of two or three

Complete the PL/pgSQL tutorial
https://www.postgresqltutorial.com/postgresql-plpgsql/

This section shows you step by step on how to use the PL/pgSQL to develop PostgreSQL user-defined functions and stored procedures.

PL/pgSQL procedural language adds many procedural elements, e.g., control structures, loops, and complex computations, to extend standard SQL. It allows you to develop complex functions and stored procedures in PostgreSQL that may not be possible using plain SQL.

PL/pgSQL procedural language is similar to the Oracle PL/SQL. The following are reasons to learn PL/pgSQL:

- PL/pgSQL is easy to learn and simple to use.
- PL/pgSQL comes with PostgreSQL by default. The user-defined functions and stored procedures developed in PL/pgSQL can be used like any built-in functions and stored procedures.
- PL/pgSQL inherits all user-defined types, functions, and operators.
- PL/pgSQL has many features that allow you to develop complex functions and stored procedures.
- PL/pgSQL can be defined to be trusted by the PostgreSQL database server.

Let's get started programming with PL/pgSQL.

Section 1. Getting started

- Introduction to PostgreSQL PL/pgSQL – introduce you to the PostgreSQL PL/pgSQL and explain to you their advantages and disadvantages.
- Dollar-quoted string constants – learn how to use dollar-quoted string constant syntax.
- Block Structure – introduce you to the PL/pgSQL block structure and show you how to develop and execute anonymous blocks.

Section 2. Variables & constants

- Variables – show you how to declare variables in PL/pgSQL.
- Select into – guide you on how to use the select into to select data and assign it to a variable.
- Row type variables – learn how to use the row variables to store a complete row of a result set.

- Record type variables – show you how to declare record variables to hold a single row of a result set.
- Constants – guide you on how to use constants to make the code more readable and easier to maintain.

Section 3. Reporting messages and errors

- Raising errors and reporting messages – show you how to report messages and raise errors in PL/pgSQL.
- Assert – show you how to use the assert statement to add debugging checks to PL/pgSQL code.

Section 4. Control structures

- If statement – introduce you to three forms of the if statement.
- Case statements – explain case statements including the simple and searched case statements.
- Loop statements – show you how to use loop statements to execute a block of code repeatedly based on a condition.
- While loop – learn how to use the while loop statement to create a pre-test loop.
- For loop – show you how to use the for loop statement to iterate over rows of a result set.
- Exit – guide you on how to use the exit statement to terminate a loop.
- Continue – provide you with a way to use the continue statement to skip the current loop iteration and start a new one.

Section 5. User-defined functions

- Create Function – show you how to develop a user-defined function by using the create function statement.
- Function parameter modes – introduce you to various parameter modes including IN, OUT, and INOUT.
- Function overloading – introduce you to the function overloading.
- Functions that return a table – show you how to develop a function that returns a table.
- Drop function – learn how to remove an existing function.

Section 6. Exception handling

- Handling exception – show you how to use the exception clause to catch and handle exceptions.

Section 7. Stored procedures

- Create procedure – show you how to create and call a stored procedure.
- Drop procedure – learn how to drop a stored procedure.

Section 8. Cursors

- Cursors – show you how to use cursors to process a result set, row by row.

Section 9. Trigger functions

- Trigger procedures using PL/pgSQL – apply PL/pgSQL to define trigger procedures.

Course: Database management
Unit: Advanced SQL 2
Assignment: Create a function
Teamwork: Groups of two or three

Create functions using all you have learned in the PL/pgSQL tutorial

You can modify the database as needed. Submit the new database together with the functions.

Write an explanatory report explaining how you have used the different things you have learned in the tutorial.

I appreciate that the functions you write are meaningful and original.

Course: Database management
Unit: Advanced SQL 2
Assignment: Present and defend your functions
Teamwork: Groups of two or three

Present and defend your functions to the rest of the class.

Course: Database management
Unit: Advanced SQL 2
Assignment: Create a cursor, a stored procedure and a trigger
Teamwork: Groups of two or three

Do the tutorials in postgresqltutorial.com related to cursors, stored procedures and triggers. Then create a cursor, a stored procedure and a trigger for our database.

Send the code you generate in a separate text file (extension .txt or .sql).

You can modify the database as needed. Submit the new database together with the functions.

Write an explanatory report explaining how you have used the different things you have learned in the tutorial.

I appreciate that the cursors, the procedures and the triggers you write are meaningful and original.

Course: Database management
Unit: Advanced SQL 2
Assignment: Present and defend your cursor, trigger, procedure
Teamwork: Groups of two or three

Present and defend your cursor, procedure and trigger to the rest of the class.

9 Unit 9: Database Design and the E-R Model

Course: Database management
Unit: Database design. The E-R model.
Material: E-R diagram tool
Teamwork: Groups of two or three

Find an E-R diagram tool that you like and familiarize yourself with it.

Some examples:
https://draw.io
https://diagrams.net
https://creately.com/

Course: Database management
Unit: Database Design. The E-R model.
Material: E-R exercises
Teamwork: Groups of two or three

Draw an ER diagram of the following cases:

1.- Hierarchical Organisation Structure.
a) A group (ID, name) has several companies (ID, name), whereas a company belongs to group.
b) Companies are connected by a hierarchical structure; each subsidiary is assigned to exactly one company of the next higher hierarchy level, the parent company.
c) Each company has several plants (ID, name, address); a plant belongs to one company only.
d) A plant produces many items (ItemNo, Description, ProductionCost). An item is only produced in one of the plants.

2.- Customer Discounts.
a) Customers (ID, name, surname) get discounts (discountNo, description, amount) on items (ItemNo, description, price). Each item can have many discount rates and a discount rate is only for a single item. A customer can have many discounts on the same item.
b) Items belong to a single category (ID, name).

3.- Products, parts and materials.
a) We have products (ID, description) formed by several parts (PartNo, description).
b) Every part is used only in a single product.
c) Products belong to a single category.

4.- Web shop.
We want to design a database for a web shop. We want to keep information about customers. We want to keep their name (composed by a first name, middle name, and last name), their date of birth, and their address. Customers will place orders of the products that we sell. Obviously, an order can contain many products (or only one). Regarding the products, we want to keep its code, its name, a description, and its price.

5.- Person in organization.
a) A person (SSN, name, address, phone_number, email) is either an employee or a manager. We want to keep relatives of a person. Employees have a salary and managers have only a commission.
b) A manager manages one department (ID, name, address) and a department may have several managers. Departments are assigned to one company (ID, name, address), while companies have several departments.
c) Employees work for one department (and a department can have many employees working in it).

6.- Webcourses.

We want to build a website to sell webcourses. Students (id_card, name, surname, e-mail, address, cell_phone) will enroll in the courses (code, name, description). The same course can be done in different dates in which students will be enrolled in it. Obviously, there are also teachers (id_card, name, surname, e-mail, address, cell_phone). Students enrolled in courses will have a single teacher assigned to it (but there can be courses without teachers).

7.- Census (=register of inhabitants).

We want to save the data of towns/villages in Spain and its inhabitants. We must consider:

- Each person can only be registered in a dwelling house (=residence).
- Each person can own more than one house and one house can have more than one owner.
- You want to store of each person his/her NIF, name, surname, and date of birth. Of the house you want to store cadastral reference and the address (street, number, floor, and door).
- Each person can depend on another person, who will be the head of the family.
- Each house is located in a town. You want to store of each town its name.
- The towns are located in regions and the regions in states. You want to store of regions/states their name.

8.- Project management.

A company wants to keep information about the departments in which it is divided and about the heads of those departments. Moreover, it must be kept data about the projects that are carried out currently, employees that participate in them and who coordinates them.
You must consider:

- Each department can only have a head and an employee can only be head of a single department.
- Each project is coordinated by a single employee who can be coordinator of several projects.
- Each employee can work in a single department.
- Each employee can work on several projects at once.
- You must select the attributes that you consider.

9.- Project management 2.

We want to design a data model of a company that is dedicated to the development of projects. These projects are ordered to the company by external customers. Each of the projects are composed of phases (with information to store).Employees working within the company will be assigned to a single phase of a project in a specific moment in time. However, throughout their professional lives each employee has worked on many phases of many projects that we want to keep information about them. The employees are college graduates with different certificates and we want to keep this information. The company is divided into departments where the employees work. Each department has a head (only one). It is considered that the heads are also employees of the company. An employee can't belong simultaneously to more than one department. You must select the attributes that you consider appropriate.

10.- Classical music concerts.

We want to collect information in a database about the classical music concert program of the Teatre Principal of Palma de Mallorca, according to the following considerations:

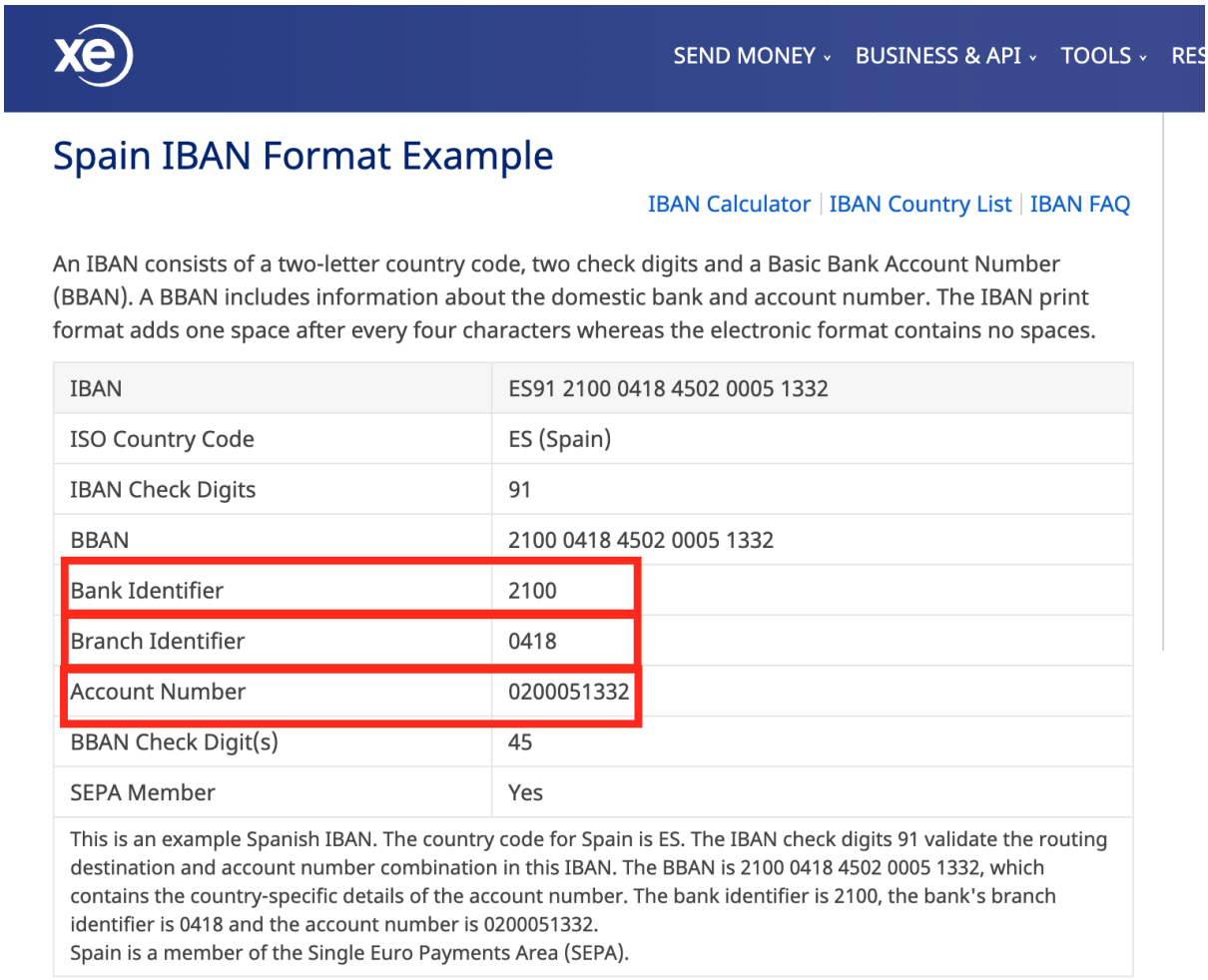
- An orchestra can perform several concerts. Only a single orchestra will perform in each concert.
- The orchestra conductor is single person (there is only a main conductor for every orchestra), but an orchestra could have several guest conductors, who could be invited in several orchestras. In the same concert several compositions can be played. Moreover, a composition can be performed in several concerts.
- In the same concert several soloists may participate.
- Each composition is written by a single composer. But we can have compositions without author...
- Obviously, the orchestra has musicians who are specialist only in a specific instrument. A musician belongs only to an orchestra. Musicians were born in different countries.
- You must select the attributes that you consider appropriate.

11.- Saving accounts.

We want to computerize the managing of savings accounts that its customers open in its branches of different banks, as well as the management of their employees. It should be considered:

- Each savings account has a number and a creation date. It is interesting to know the movements with their amount. A movement is a transfer of an amount of money from a savings account to other savings account. There will be different types of movements.
- The customer's NIF, name, surname, address and telephone number are stored.
- An account in a branch has a single owner, that is, it is opened by a single owner. Shared accounts are not allowed.
- Each branch is known by its branch identifier, its address and its telephone number.
- The employees are exclusively assigned to a branch. They want to know their NIF, name, surname, address and salary of the employees.
- Each branch has a branch director and they want to know their name, surname, address and salary.
- Employees (and directors) may change the branch where they work (we want to know the start and end date in every branch).
- Banks: We need to know their identifier and their name.

Clue, bank account format:



SEND MONEY BUSINESS & API TOOLS RES

Spain IBAN Format Example

IBAN Calculator | IBAN Country List | IBAN FAQ

An IBAN consists of a two-letter country code, two check digits and a Basic Bank Account Number (BBAN). A BBAN includes information about the domestic bank and account number. The IBAN print format adds one space after every four characters whereas the electronic format contains no spaces.

IBAN	ES91 2100 0418 4502 0005 1332
ISO Country Code	ES (Spain)
IBAN Check Digits	91
BBAN	2100 0418 4502 0005 1332
Bank Identifier	2100
Branch Identifier	0418
Account Number	0200051332
BBAN Check Digit(s)	45
SEPA Member	Yes

This is an example Spanish IBAN. The country code for Spain is ES. The IBAN check digits 91 validate the routing destination and account number combination in this IBAN. The BBAN is 2100 0418 4502 0005 1332, which contains the country-specific details of the account number. The bank identifier is 2100, the bank's branch identifier is 0418 and the account number is 0200051332.
Spain is a member of the Single Euro Payments Area (SEPA).

12.- Real estate agency.

A real estate agency, with several branches, wants to computerize the sales of parcels of land. The information needed:

- The owners transfer their parcels to a branch (only to one) of the agency. An owner can have several parcels of land transferred to different branches.
- The parcels transferred to the estate agency are sold to customers through a certain salesman. The owners want a minimum price for the parcel.
- There are two types of salesman in the company: those who work permanently in a certain branch of the company (with a fixed salary) and those who do so sporadically in one or more branches and who only earn a commission based on sales. The sellers who work permanently in a branch have commission for sales also.

The data we must store is the following:

- Owners: NIF, name, surname, address and telephone number.
- Parcels: Cadastral_number, location.
- Customers: NIF, name, surname, date of birth, address and telephone number.
- Salesmen: NIF, name, surnames and telephone.
- Branches: Address, telephone.

Finally, they want to keep information on the amount and date of sales that are made.
VERY IMPORTANT: A parcel can be sold or transferred **only one time** (that means that if we sell the parcel for a second time it will be introduced in the system as a new one).

13.- Crusades in the Middle Age.
A professor of medieval history wants a database. The specifications are:

- Knights: It is interesting to keep their name, their date of birth, and their father (according to the social rules of their time, knights can only be children of another knight). It is also interesting to know in what province they were born, in what province they governed and in what crusades they participated under the king's orders. Finally, we want to keep —if any— a quote claimed by the knight in combat (e.g. “For God’s glory!”).
- Provinces: It is interesting to know their name. In addition, we want to know the knights who have governed them and those who were born there. A province can be governed by several knights on different dates and a knight can govern in several provinces (on different dates). It is interesting to know the date of beginning and end of government of each knight on the province. Provinces are part of a kingdom.
- Crusades: It is interesting to know its name, start date, end date, against whom was the crusade (a kingdom) and its result, as well as the kings and knights participating in it. Keep in mind that more than one king can participate in each crusade.
- Kings: You want to store data corresponding to their names, their date of birth, start and end dates of their reign and the kingdoms over which they reigned. Finally, we want to know the main color of their dynasty.

14.- Project management 3.
A company wants to design a database to store in it all the information generated in the projects that carries out.
For each projects carried out, it is important to store the code, description, total cost of the project, estimated cost of the project, start date and end date. The projects are developed for customers that they wish to keep the code, telephone, address and company name. A customer can have several projects, but a single project is only for a single customer.
There are also collaborators involved in projects. Regarding collaborators, we want to keep: nif, name, address, telephone and account number. A collaborator can participate in several projects. The projects are developed by one or more collaborators.
Collaborators of the projects receive payments. Regarding the payments made, they want to keep the payment number, concept, amount and date of payment. It is also interesting to store the different types of payments that the company can make. For each of the types of payments they want to keep code and description. A type of payment can belong to several payments.

15.- Travel agency.
A travel agency wants to computerize all the management of travelers who come to the agency and the trips they make.The agency gave us with the following information:

- The agency wants to keep the following travellers’ data: ID, name, address and telephone number.
- For each trip handled by the agency, it is important to keep the trip code, maximum number of travelers, start date of the trip is made and for how many days. A traveler can make as many trips as she/he wants with the agency.

- Each trip made has a destination and a place of origin. From each of them, they want to store the code, name and other information that may be of interest. A trip has a unique place of destination and a unique place of origin.

16.- Booking a hotel.

The aim is to design a database in the E -R model for a hotel company:

- Each hotel (you want to keep: name, address, telephone number, year of construction, etc.) is necessarily classified in one category (for example, three stars) and can be downgraded or upgraded in a concrete date.
- Each category has various information associated with it, such as, for example, the type of VAT that corresponds to it and the description.
- Hotels have different kinds of rooms (suites, doubles, singles, etc.), they define the number of hosts in the room (you also want to keep a code, a name, and a description). The rooms are numbered and you want to know the floor where the rooms are located. For each room you want to save their code and the type of room. Every room has a price.
- Customers can book hotel rooms. The id_card (passport or country id card), name, address and telephone number will appear in their private booking information. Date_of_birth, Obviously, a booking has a check-in date and a check-out date.
- Travel agencies can also book rooms. In case a booking is made by a travel agency, the same data as for private customers will be needed, in addition to the name of the person of the travel agency who is making the reservation.

17.- Accidents and fines.

Imagine that an insurance company in your town asks you for a database to keep track of accidents and fines. After some interviews with your new customer, you have the following information:

- You must store information of all the people who have a vehicle. It is necessary to keep the personal data of each person (name, surname, address, town, telephone and ID).
- For each vehicle you want to store the license plate (or number plate), the brand, model, and color. A person may have several vehicles, and it may be possible that a vehicle belongs to several persons at the same time.
- It is also desired to keep information to manage the traffic accidents of the province. Each traffic accident has a correlative reference number according to the order of entry to the database. You want to know the date, place and time that each accident took place. It could be possible that an accident may involve several people and several vehicles.
- You also want to keep a record of the fines of your customers. Each fine will be assigned a correlative reference number. In addition, the date, time, place, description and amount there must be stored. A fine will only apply to a driver and involves only one vehicle.

18.- Drugstore.

A drugstore (chemist's or pharmacy) wants a computer system to manage the stock, location, sales and orderings of drugs (=medications).

The queries planned are:

- Stock, location (shelf), and price of a drug.
- Drugs of a certain drug company (provider). We will suppose that we buy drugs directly to the drug company that produces them.
- List of drugs that are below the stock (that has been set to minimum number of units).
- Pre-order list, with date: for each drug company, the list of drugs that must be ordered together with their number of units because they are below the minimum stock.

Keep in mind that:

- The client sometimes don't know the name of the drug, but they know the applications of the drug. For example, headache, stomach pain, toothache, ear pain, antivirals, etc. Obviously, the same drug may have several applications. The client sometimes also knows the drug company that distributes the medication.
- A drug is only distributed by a drug company.
- A drug is never stored on several shelves.
- The cash registers have a barcode reader.

In a short time, they will want to expand the system with new features such as:

- Statistics on drugs sold, sales per day, profitability by medication, profitability by laboratory,

It would be advisable that appropriate data will be stored to offer these outputs in the future.

19.- Airline company.

An airline company wants to build a database that can be accessed through terminals from the stands opened to the public in different airports. Reservations can only be done from the stands and reservations are only for a single flight. In particular, you want to keep information about what they call "generic flights" (flight number, departure airport and destination airport, departure and arrival dates, and departure and arrival times) and the state of the reservations of seats (number of free seats) for a certain flight of a certain day. All flights are daily and direct (without scales). There may be more than one daily flight between the same cities, at different times (but only one at the same time). The most typical queries will be:

- Given cities of origin and destination, list all flights between each other (flight number and time).
- Given a flight number and a date, list the number of free seats.

20.- Products and commissions.

A database for a small business must contain information about customers, items and orders. So far, the following data is stored in several documents:

- For each customer: Customer number (unique), shipping addresses (many per customer), balance (="*saldo*"), credit limit (depends on the customer, but it can't exceed 3000€), discount.
- For each item: Item number (unique), factories that distribute it, stock of that item in each factory, description of the item.
- For each order: Each order has a header and the body of the order. The header is formed by the customer number, shipping address and date of the order. The body of

the order is divided in several lines, in each line the number of line, units of the product and price.

In addition, it has been determined that data about the factories should be stored: Factory number and telephone number. And you want to see how many items (in total) the factory provides.

Note: An address will be understood as number, street, city and province. A date includes time.

21.- Olympics.

The Olympic host cities are divided into sports complexes. The sports complexes are subdivided into those in which a single sport is developed. The sports complexes have areas for each sport with a location indicator (example: center, cornerNE, etc.). A complex has a location, a head of organization and a total occupied area. Each complex holds a series of events (example: the stadium running track can hold many different races). For each event there is a date, hour, duration, number of participants, number of commissioners. A list of all commissioners will be kept together with the list of events in which each commissioner is involved (with the task of judge or observer). Finally, for each event and for maintenance, some equipment will be needed (example: arcs, poles, parallel bars, etc).

Course: Database management
Unit: Database Design. The E-R model.
Assignment: Create an ERD
Teamwork: Groups of two or three

Work in pairs. Pick one of the exercises or create your own description of a database. Then, draw the Entity Relation Diagram. Prepare an explanatory report.

For extra points,

- Use composite attributes, multi-valued attributes and calculated (derived) attributes.
- Use 1:1, 1:N and M:N relationships.
- Represent minimum and maximum cardinalities.
- With and without total participation.
- Weak entities.
- Specialization/Generalization
- Total generalization
- Disjoint generalization/ Overlapping generalization
- Draw the same diagram in mysql workbench

Course: Database management
Unit: Database Design. The E-R model.
Assignment: Defend your ERD
Teamwork: Groups of two or three

Present your ERD and answer the questions of your classmates and your teacher.

10 Unit 10: Database Design. E-R Model to Relational Model

Course: Database management
Unit: Database Design. E-R model to relational model
Material: E-R to relational exercises
Teamwork: Groups of two or three

In the previous lesson you created ER diagrams for the following cases. Now convert those ER diagrams to relational diagrams.

1.- Hierarchical Organisation Structure.
a) A group (ID, name) has several companies (ID, name), whereas a company belongs to group.
b) Companies are connected by a hierarchical structure; each subsidiary is assigned to exactly one company of the next higher hierarchy level, the parent company.
c) Each company has several plants (ID, name, address); a plant belongs to one company only.
d) A plant produces many items (ItemNo, Description, ProductionCost). An item is only produced in one of the plants.

2.- Customer Discounts.
a) Customers (ID, name, surname) get discounts (discountNo, description, amount) on items (ItemNo, description, price). Each item can have many discount rates and a discount rate is only for a single item. A customer can have many discounts on the same item.
b) Items belong to a single category (ID, name).

3.- Products, parts and materials.
a) We have products (ID, description) formed by several parts (PartNo, description).
b) Every part is used only in a single product.
c) Products belong to a single category.

4.- Web shop.
We want to design a database for a web shop. We want to keep information about customers. We want to keep their name (composed by a first name, middle name, and last name), their date of birth, and their address. Customers will place orders of the products that we sell. Obviously, an order can contain many products (or only one). Regarding the products, we want to keep its code, its name, a description, and its price.

5.- Person in organization.
a) A person (SSN, name, address, phone_number, email) is either an employee or a manager. We want to keep relatives of a person. Employees have a salary and managers have only a commission.
b) A manager manages one department (ID, name, address) and a department may have several managers. Departments are assigned to one company (ID, name, address), while companies have several departments.
c) Employees work for one department (and a department can have many employees working in it).

6.- Webcourses.
We want to build a website to sell webcourses. Students (id_card, name, surname, e-mail, address, cell_phone) will enroll in the courses (code, name, description). The same course can be done in different dates in which students will be enrolled in it. Obviously, there are also teachers (id_card, name, surname, e-mail, address, cell_phone). Students enrolled in courses will have a single teacher assigned to it (but there can be courses without teachers).

7.- Census (=register of inhabitants).
We want to save the data of towns/villages in Spain and its inhabitants. We must consider:

- Each person can only be registered in a dwelling house (=residence).
- Each person can own more than one house and one house can have more than one owner.
- You want to store of each person his/her NIF, name, surname, and date of birth. Of the house you want to store cadastral reference and the address (street, number, floor, and door).
- Each person can depend on another person, who will be the head of the family.
- Each house is located in a town. You want to store of each town its name.
- The towns are located in regions and the regions in states. You want to store of regions/states their name.

8.- Project management.
A company wants to keep information about the departments in which it is divided and about the heads of those departments. Moreover, it must be kept data about the projects that are carried out currently, employees that participate in them and who coordinates them.
You must consider:

- Each department can only have a head and an employee can only be head of a single department.
- Each project is coordinated by a single employee who can be coordinator of several projects.
- Each employee can work in a single department.
- Each employee can work on several projects at once.
- You must select the attributes that you consider.

9.- Project management 2.
We want to design a data model of a company that is dedicated to the development of projects. These projects are ordered to the company by external customers. Each of the projects are composed of phases (with information to store).Employees working within the company will be assigned to a single phase of a project in a specific moment in time. However, throughout their professional lives each employee has worked on many phases of many projects that we want to keep information about them. The employees are college graduates with different certificates and we want to keep this information. The company is divided into departments where the employees work. Each department has a head (only one). It is considered that the heads are also employees of the company. An employee can't belong simultaneously to more than one department. You must select the attributes that you consider appropriate.

10.- Classical music concerts.

We want to collect information in a database about the classical music concert program of the Teatre Principal of Palma de Mallorca, according to the following considerations:

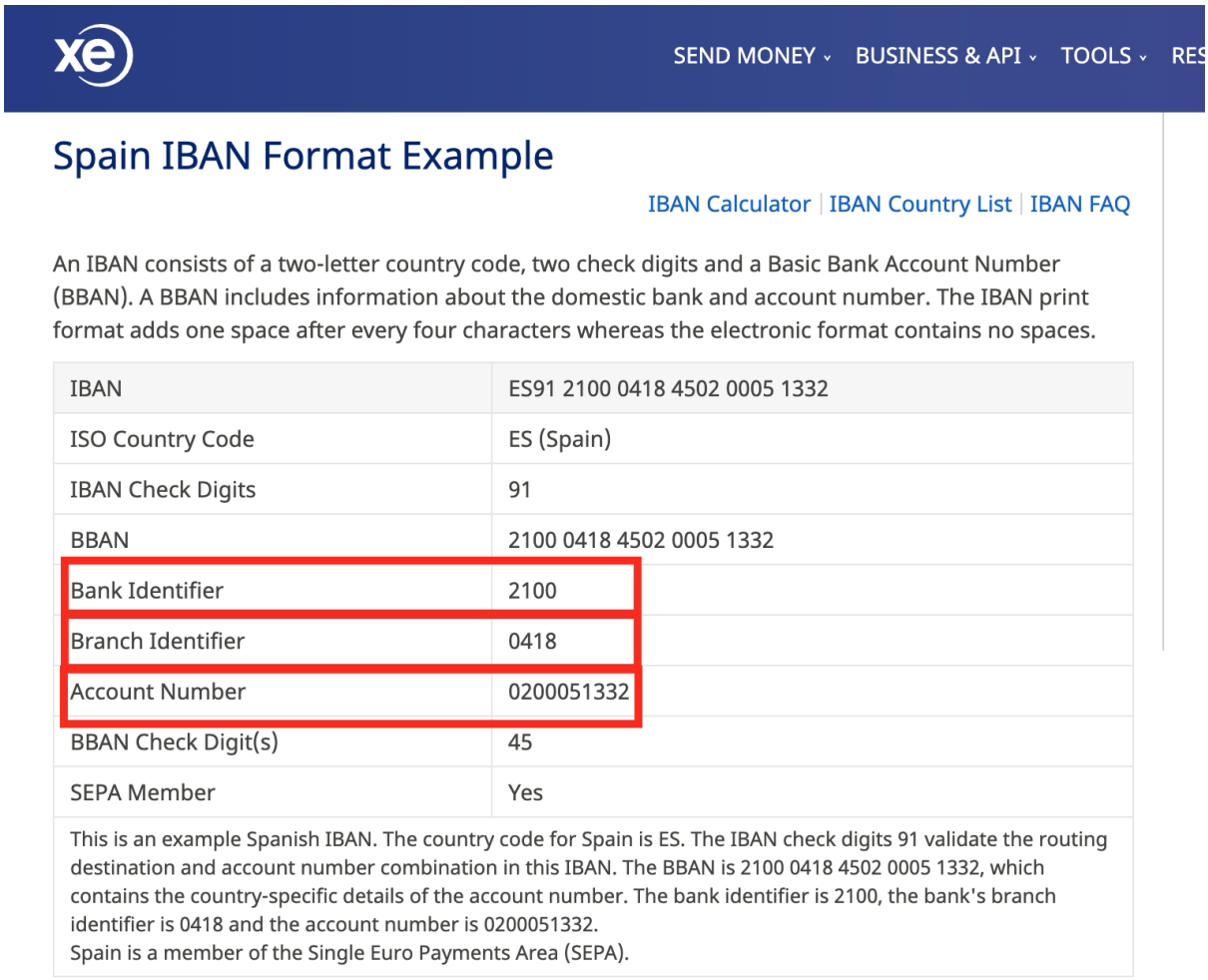
- An orchestra can perform several concerts. Only a single orchestra will perform in each concert.
- The orchestra conductor is single person (there is only a main conductor for every orchestra), but an orchestra could have several guest conductors, who could be invited in several orchestras. In the same concert several compositions can be played. Moreover, a composition can be performed in several concerts.
- In the same concert several soloists may participate.
- Each composition is written by a single composer. But we can have compositions without author...
- Obviously, the orchestra has musicians who are specialist only in a specific instrument. A musician belongs only to an orchestra. Musicians were born in different countries.
- You must select the attributes that you consider appropriate.

11.- Saving accounts.

We want to computerize the managing of savings accounts that its customers open in its branches of different banks, as well as the management of their employees. It should be considered:

- Each savings account has a number and a creation date. It is interesting to know the movements with their amount. A movement is a transfer of an amount of money from a savings account to other savings account. There will be different types of movements.
- The customer's NIF, name, surname, address and telephone number are stored.
- An account in a branch has a single owner, that is, it is opened by a single owner. Shared accounts are not allowed.
- Each branch is known by its branch identifier, its address and its telephone number.
- The employees are exclusively assigned to a branch. They want to know their NIF, name, surname, address and salary of the employees.
- Each branch has a branch director and they want to know their name, surname, address and salary.
- Employees (and directors) may change the branch where they work (we want to know the start and end date in every branch).
- Banks: We need to know their identifier and their name.

Clue, bank account format:



SEND MONEY BUSINESS & API TOOLS RES

Spain IBAN Format Example

IBAN Calculator | IBAN Country List | IBAN FAQ

An IBAN consists of a two-letter country code, two check digits and a Basic Bank Account Number (BBAN). A BBAN includes information about the domestic bank and account number. The IBAN print format adds one space after every four characters whereas the electronic format contains no spaces.

IBAN	ES91 2100 0418 4502 0005 1332
ISO Country Code	ES (Spain)
IBAN Check Digits	91
BBAN	2100 0418 4502 0005 1332
Bank Identifier	2100
Branch Identifier	0418
Account Number	0200051332
BBAN Check Digit(s)	45
SEPA Member	Yes

This is an example Spanish IBAN. The country code for Spain is ES. The IBAN check digits 91 validate the routing destination and account number combination in this IBAN. The BBAN is 2100 0418 4502 0005 1332, which contains the country-specific details of the account number. The bank identifier is 2100, the bank's branch identifier is 0418 and the account number is 0200051332.
Spain is a member of the Single Euro Payments Area (SEPA).

12.- Real estate agency.

A real estate agency, with several branches, wants to computerize the sales of parcels of land. The information needed:

- The owners transfer their parcels to a branch (only to one) of the agency. An owner can have several parcels of land transferred to different branches.
- The parcels transferred to the estate agency are sold to customers through a certain salesman. The owners want a minimum price for the parcel.
- There are two types of salesman in the company: those who work permanently in a certain branch of the company (with a fixed salary) and those who do so sporadically in one or more branches and who only earn a commission based on sales. The sellers who work permanently in a branch have commission for sales also.

The data we must store is the following:

- Owners: NIF, name, surname, address and telephone number.
- Parcels: Cadastral_number, location.
- Customers: NIF, name, surname, date of birth, address and telephone number.
- Salesmen: NIF, name, surnames and telephone.
- Branches: Address, telephone.

Finally, they want to keep information on the amount and date of sales that are made.
VERY IMPORTANT: A parcel can be sold or transferred **only one time** (that means that if we sell the parcel for a second time it will be introduced in the system as a new one).

13.- Crusades in the Middle Age.
A professor of medieval history wants a database. The specifications are:

- Knights: It is interesting to keep their name, their date of birth, and their father (according to the social rules of their time, knights can only be children of another knight). It is also interesting to know in what province they were born, in what province they governed and in what crusades they participated under the king's orders. Finally, we want to keep —if any— a quote claimed by the knight in combat (e.g. “For God’s glory!”).
- Provinces: It is interesting to know their name. In addition, we want to know the knights who have governed them and those who were born there. A province can be governed by several knights on different dates and a knight can govern in several provinces (on different dates). It is interesting to know the date of beginning and end of government of each knight on the province. Provinces are part of a kingdom.
- Crusades: It is interesting to know its name, start date, end date, against whom was the crusade (a kingdom) and its result, as well as the kings and knights participating in it. Keep in mind that more than one king can participate in each crusade.
- Kings: You want to store data corresponding to their names, their date of birth, start and end dates of their reign and the kingdoms over which they reigned. Finally, we want to know the main color of their dynasty.

14.- Project management 3.
A company wants to design a database to store in it all the information generated in the projects that carries out.
For each projects carried out, it is important to store the code, description, total cost of the project, estimated cost of the project, start date and end date. The projects are developed for customers that they wish to keep the code, telephone, address and company name. A customer can have several projects, but a single project is only for a single customer.
There are also collaborators involved in projects. Regarding collaborators, we want to keep: nif, name, address, telephone and account number. A collaborator can participate in several projects. The projects are developed by one or more collaborators.
Collaborators of the projects receive payments. Regarding the payments made, they want to keep the payment number, concept, amount and date of payment. It is also interesting to store the different types of payments that the company can make. For each of the types of payments they want to keep code and description. A type of payment can belong to several payments.

15.- Travel agency.
A travel agency wants to computerize all the management of travelers who come to the agency and the trips they make.The agency gave us with the following information:

- The agency wants to keep the following travellers’ data: ID, name, address and telephone number.
- For each trip handled by the agency, it is important to keep the trip code, maximum number of travelers, start date of the trip is made and for how many days. A traveler can make as many trips as she/he wants with the agency.

- Each trip made has a destination and a place of origin. From each of them, they want to store the code, name and other information that may be of interest. A trip has a unique place of destination and a unique place of origin.

16.- Booking a hotel.

The aim is to design a database in the E -R model for a hotel company:

- Each hotel (you want to keep: name, address, telephone number, year of construction, etc.) is necessarily classified in one category (for example, three stars) and can be downgraded or upgraded in a concrete date.
- Each category has various information associated with it, such as, for example, the type of VAT that corresponds to it and the description.
- Hotels have different kinds of rooms (suites, doubles, singles, etc.), they define the number of hosts in the room (you also want to keep a code, a name, and a description). The rooms are numbered and you want to know the floor where the rooms are located. For each room you want to save their code and the type of room. Every room has a price.
- Customers can book hotel rooms. The id_card (passport or country id card), name, address and telephone number will appear in their private booking information. Date_of_birth, Obviously, a booking has a check-in date and a check-out date.
- Travel agencies can also book rooms. In case a booking is made by a travel agency, the same data as for private customers will be needed, in addition to the name of the person of the travel agency who is making the reservation.

17.- Accidents and fines.

Imagine that an insurance company in your town asks you for a database to keep track of accidents and fines. After some interviews with your new customer, you have the following information:

- You must store information of all the people who have a vehicle. It is necessary to keep the personal data of each person (name, surname, address, town, telephone and ID).
- For each vehicle you want to store the license plate (or number plate), the brand, model, and color. A person may have several vehicles, and it may be possible that a vehicle belongs to several persons at the same time.
- It is also desired to keep information to manage the traffic accidents of the province. Each traffic accident has a correlative reference number according to the order of entry to the database. You want to know the date, place and time that each accident took place. It could be possible that an accident may involve several people and several vehicles.
- You also want to keep a record of the fines of your customers. Each fine will be assigned a correlative reference number. In addition, the date, time, place, description and amount there must be stored. A fine will only apply to a driver and involves only one vehicle.

18.- Drugstore.

A drugstore (chemist's or pharmacy) wants a computer system to manage the stock, location, sales and orderings of drugs (=medications).

The queries planned are:

- Stock, location (shelf), and price of a drug.
- Drugs of a certain drug company (provider). We will suppose that we buy drugs directly to the drug company that produces them.
- List of drugs that are below the stock (that has been set to minimum number of units).
- Pre-order list, with date: for each drug company, the list of drugs that must be ordered together with their number of units because they are below the minimum stock.

Keep in mind that:

- The client sometimes don't know the name of the drug, but they know the applications of the drug. For example, headache, stomach pain, toothache, ear pain, antivirals, etc. Obviously, the same drug may have several applications. The client sometimes also knows the drug company that distributes the medication.
- A drug is only distributed by a drug company.
- A drug is never stored on several shelves.
- The cash registers have a barcode reader.

In a short time, they will want to expand the system with new features such as:

- Statistics on drugs sold, sales per day, profitability by medication, profitability by laboratory,

It would be advisable that appropriate data will be stored to offer these outputs in the future.

19.- Airline company.

An airline company wants to build a database that can be accessed through terminals from the stands opened to the public in different airports. Reservations can only be done from the stands and reservations are only for a single flight. In particular, you want to keep information about what they call "generic flights" (flight number, departure airport and destination airport, departure and arrival dates, and departure and arrival times) and the state of the reservations of seats (number of free seats) for a certain flight of a certain day. All flights are daily and direct (without scales). There may be more than one daily flight between the same cities, at different times (but only one at the same time). The most typical queries will be:

- Given cities of origin and destination, list all flights between each other (flight number and time).
- Given a flight number and a date, list the number of free seats.

20.- Products and commissions.

A database for a small business must contain information about customers, items and orders. So far, the following data is stored in several documents:

- For each customer: Customer number (unique), shipping addresses (many per customer), balance (="*saldo*"), credit limit (depends on the customer, but it can't exceed 3000€), discount.
- For each item: Item number (unique), factories that distribute it, stock of that item in each factory, description of the item.
- For each order: Each order has a header and the body of the order. The header is formed by the customer number, shipping address and date of the order. The body of

the order is divided in several lines, in each line the number of line, units of the product and price.

In addition, it has been determined that data about the factories should be stored: Factory number and telephone number. And you want to see how many items (in total) the factory provides.

Note: An address will be understood as number, street, city and province. A date includes time.

21.- Olympics.

The Olympic host cities are divided into sports complexes. The sports complexes are subdivided into those in which a single sport is developed. The sports complexes have areas for each sport with a location indicator (example: center, cornerNE, etc.). A complex has a location, a head of organization and a total occupied area. Each complex holds a series of events (example: the stadium running track can hold many different races). For each event there is a date, hour, duration, number of participants, number of commissioners. A list of all commissioners will be kept together with the list of events in which each commissioner is involved (with the task of judge or observer). Finally, for each event and for maintenance, some equipment will be needed (example: arcs, poles, parallel bars, etc).

Course: Database management
Unit: Database Design. E-R model to relational model
Assignment: ERM to relational model
Teamwork: Groups of two or three

Convert your ERD of the previous lesson to a relational model.

You can represent the relational model diagrammatically, as a mysql workbench diagram, as a list of relations indicating PK and FK and/or as a SQL script.

Use sensible names.

Ideally your database should have:

- Composite attributes, multi-valued attributes and calculated (derived) attributes.
- 1:1, 1:N and M:N relationships.
- Relationships with and without total participation.
- Weak entities.
- Specialization/Generalization
- Total/Partial generalization
- Disjoint generalization/ Overlapping generalization

If your initial ERD did not include all the items of the checklist above, you can modify your ERD to include them before transforming it to a relational model.

Document those constraints that cannot be captured in the logical design.

Create the database in a RDBMS of your choice and insert a set of data to verify that the implementation conforms to the model.

Document the data dictionary.

As usual, deliver an explanatory report of your work.

Course: Database management
Unit: Database Design. E-R model to relational model
Assignment: Defend your ERM to relational model
Teamwork: Groups of two or three

Present your Relational Model to your classmates and answer the questions of the teacher and the other classmates..

11 Unit 11: Database Design. Normalization

U11 Normalization

Jaume Barceló

CIFP Francesc de Borja Moll

Database Management

Contents

Introduction

- If we derive the relational model from an Entity-Relationship Model, most of the times it will be correct.
- There are a number of normalization rules that help us to improve the design of our database.
- We will see, with examples, those that are easier and more useful: NF1, NF2, NF3, BCNF

Functional Dependency

- Given a relation R, a set of attributes X in R is said to functionally determine another set of attributes Y, also in R, (written $X \xrightarrow{FD} Y$) if, and only if, each X value in R is associated with precisely one Y value in R; R is then said to satisfy the functional dependency $X \xrightarrow{FD} Y$.
- customer(customer_id [PK], name, address, phone)
- stock(spare[PK], warehouse[PK], quantity)

Fully Functional Dependency

- Given a relation R, a set of attributes X in R is said to fully functionally determine another set of attributes Y, also in R, (written $X \xrightarrow{FFD} Y$) if, and only if,
- Y depends functionally on X and
- Y does not functionally depend on any proper subset of X.
- customer(customer_id [PK], name, address, phone)
- stock(spare[PK], warehouse[PK], quantity)

First Normal Form

- The domain of each attribute contains only atomic (indivisible) values (the value of each attribute contains only a single value from that domain).
- In other words, we have to get rid of composite atributes and multi-valued attributes.

2NF and 3NF Based on Primary Keys

- When defining the 2NF and the 3NF, it is possible to do so based on primary keys or in a more general form considering all candidate keys.
- For simplicity, we will consider definitions based on primary keys.

Second Normal Form

- It is 1NF
- Every nonprime attribute A in R is fully functionally dependent on the primary key of R.
- purchase(supplier[PK], article[PK], date[PK], quantity, city_supplier)

Second Normal Form (II)

- supplier(supplier[PK], city)
- purchase(supplier[PK][FK], article[PK], date[PK], quantity)

Third Normal Form

- It is 2NF and
- No nonprime attribute of R is transitively dependent on the primary key.
- employee(employee_id[PK], department_id, department_location)

Third Normal Form II

- employee(employee_id[PK], department_id[FK])
- department(department_id[PK], department_location)

Boyce-Codd Normal Form

- It is in 3NF
- For each candidate key it is verified that each attribute that does not belong to it is fully functionally dependent on it.
- school(student_id[PK], course_id[PK], professor_id)
- student_id, course_id $\xrightarrow{FFD}$ professor_id
- professor_id $\rightarrow$ course_id

Boyce-Codd Normal Form (II)

- It is not BCNF because there is a candidate key (student_id, professor_id) that results in the third columm (course_id) not being fully functional dependant on the candidate key, as the course depends only on the professor.

Boyce-Codd Normal Form (III)

- school_professor(professor_id[PK], course_id)
- school_student(student_id[PK], professor_id[PK][FK])

Course: Database management
Unit: Database Design. Normalization
Material: Normalization exercises
Teamwork: Groups of two or three

To do the exercises remember that:

- 1NF: The domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.
- 2NF: 1NF + any attribute that is not part of the primary key depends fully functionally on the entire primary key.
- 3NF: 2NF + each non-key attribute depends only on the primary key.

1.- For the example below we have one big table. Put the table in normalized form.

SID = Student ID, S_Name= Student Name,

CID = Course ID, C_Name = Course Name, Grade = Student's Grade in Course

Faculty = Faculty Name, F_Phone = Faculty Phone

Functional Dependencies are:

SID → S_name SID and CID → Grade CID → C_name

CID → Faculty Faculty → F_phone

The primary key of the table is SID.

SID	CID	S_name	C_name	Grade	Faculty	F_phone
1	IS318, IS301	Adams	Database, EC	A,B	Howser, Langley	60192, 45869
2	IS318	Jones	Database	A	Howser	60192
3	IS318	Smith	Database	B	Howser	60192
4	IS301, IS318	Baker	EC, Database	A,B	Langley, Howser	45869, 60192

Put the above table in 1NF Tables:

Put the above table in 2NF:

Put the above table in 3NF Tables:

Final set of Tables with meaningful names and PKs and FKs:

2.- For the example below we have one big table. Put the table in normalized form.

OID = Order ID, O_Date= Order Date,

CID = Customer ID, C_Name = Customer Name, C_State = Customer's State,

PID = product id, P_Desc =Product Name, P_Price = Product Price, Qty = Quantity
Purchased Note: 7, 5, 4 means three Product IDs. Similarly, 1, 1, 5 means three Quantities.

Functional Dependencies are:

OID → O_Date CID → C_Name PID → P_Desc PID → P_Price

OID → CID CID → C_State PID and OID → Qty

OID	O_Date	CID	C_Name	C_State	PID	P_Desc	P_Price	Qty
1006	10/24/09	2	Apex	NC	7, 5, 4	Table, Desk, Chair	800, 325, 200	1, 1, 5
1007	10/25/09	6	Acme	GA	11, 4	Dresser, Chair	500, 200	4, 6

The primary key of the table is OID.

Put the above table in 1NF Tables:

Put the above table in 2NF:

Put the above table in 3NF Tables:

Final set of Tables with meaningful names and PKs and FKs:

3.- For the example below we have one big table representing a company's data on their projects and employees. Put the table in normalized form.

DID = Department ID, Dname = Department Name

EID = Employee ID, Ename = Employee Name, Btime = Budgeted Time

PID = Project ID, Pname = Project Name

Functional Dependencies are:

DID → Dname EID → Ename EID and PID → Btime

EID → DID PID → Pname

DID	Dname	EID	Ename	PID	Pname	Btime
10	Finance	1, 5, 11	Huey, Dewey, Louie	27, 25, 22	Alpha, Beta, Gamma	4.5, 3, 7
14	R&D	2, 4,	Jack, Jill	26, 21	Pail, Hill	8, 9

Put the above table in 1NF Tables:

Put the above table in 2NF:

Put the above table in 3NF Tables:

Final set of Tables with meaningful names and PKs and FKs:

4.-
Produce the Third Normal Form of this document by normalization.

Order Form

Order number: 1234 Date: 11/04/98

Customer number: 9876

Customer name: Billy

Customer address: 456 HighTower Street

City-Country: Hong Kong, China

ProductNo	Desscription	Quantity	Unit Price
A123	Pencil	100	$3.00
B234	Eraser	200	$1.50
C345	Sharpener	5	$8.00

Order number = OID, Customer number = CID, Customer name = C_name, Customer address = C_address, ProductNo = PID, Description = P_desc, Quantity = Qty, Unit Price = P_Price

Functional Dependencies are:

OID → Date CID → C_Name CID → City CID → C_address

PID → P_Desc PID → P_Price OID → CID City → Country

PID and OID → Qty

Put the above table in 1NF Tables:

Put the above table in 2NF:

Put the above table in 3NF Tables:

Final set of Tables with meaningful names and PKs and FKs:

5.- Do the relational model in 3NF:

PET ID	PET NAME	PET TYPE	PET AGE	OWNER	VISIT DATE	PROCEDURE
246	ROVER	DOG	12	SAM COOK	JAN 13/2002	01 - RABIES VACCINATION
					MAR 27/2002	10 - EXAMINE and TREAT WOUND
					APR 02/2002	05 - HEART WORM TEST
298	SPOT	DOG	2	TERRY KIM	JAN 21/2002	08 - TETANUS VACCINATION
					MAR 10/2002	05 - HEART WORM TEST
341	MORRIS	CAT	4	SAM COOK	JAN 23/2001	01 - RABIES VACCINATION
					JAN 13/2002	01 - RABIES VACCINATION
519	TWEEDY	BIRD	2	TERRY KIM	APR 30/2002	20 - ANNUAL CHECK UP
					APR 30/2002	12 - EYE WASH

6.- Do the relational model in 3NF:

INVOICE

HILLTOP ANIMAL HOSPITAL
INVOICE # 987

DATE: JAN 13/2002

MR. RICHARD COOK
123 THIS STREET
MY CITY, ONTARIO
Z5Z 6G6

PET	PROCEDURE	AMOUNT
ROVER	RABIES VACCINATION	30.00
MORRIS	RABIES VACCINATION	24.00
	TOTAL	54.00
	TAX (8%)	4.32
	AMOUNT OWING	58.32

7.- Do the relational model in 3NF:

Gallery Customer History Form

Gallery Customer History Form

Customer Name

Jackson, Elizabeth Phone (206) 284-6783
123 – 4th Avenue
Fonthill, ON
L3J 4S4

Purchases Made

Artist	Title	Purchase Date	Sales Price
03 - Carol Channing	Laugh with Teeth	09/17/2000	7000.00
15 - Dennis Frings	South toward Emerald Sea	05/11/2000	1800.00
03 - Carol Channing	At the Movies	02/14/2002	5550.00
15 - Dennis Frings	South toward Emerald Sea	07/15/2003	2200.00

The Gill Art Gallery wishes to maintain data on their customers, artists and paintings. They may have several paintings by each artist in the gallery at one time. Paintings may be bought and sold several times. In other words, the gallery may sell a painting, then buy it back at a later date and sell it to another customer.

8.- Do the relational model in 3NF:

Good News Grocers

User View 1 - Price Update List

Department	**Product Code**	**Aisle[1] Number**	**Price**	**Unit of Measure**
Produce	4081	1	0.35	lb
Produce	4027	1	0.90	ea
Produce	4108	1	1.99	lb
Butcher	331100	5	1.50	lb
Butcher	331105	5	2.40	lb

[1] http://www.wordreference.com/es/translation.asp?tranword=aisle

Butcher	332110	5	5.00	lb
Freezer	411100	6	1.00	ea
Freezer	521101	6	1.00	ea
Freezer	866503	6	5.00	ea
Freezer	866504	6	5.00	ea

This report is used by the department managers to update the prices that are displayed in the grocery store for these products.

9.- IESFBMOLL Computing wants to computerize their invoice system:

IESFBMOLL Computing					
INVOICE				Number: 1254	
				Date: 3/10/2018	
Customer NIF:	11111111-A				
Customer name:	Sergio González Rubio				
Address:	Caracas Street, 6				
Postal Code:	07006				
Town:	Palma				
Phone:	634216316				
Item List					
Category	**Code**	**Description**	**Price/Unit**	**Quantity**	**Price**
RAM memmories	123	Samsung DDR3-1066 SO-DIMM (4 GB, 204 pines, PC3-8500)	34,95 €	2	69,90 €
RAM memmories	124	Crucial CT102464BF160B - Memoria RAM de 8 GB (DDR3L, 1600 MT/s, PC3L-12800, SODIMM, 204-Pin)	54,99 €	3	164,97 €
RAM memmories	215	Kingston KVR13N9S8K2/8 - Memoria RA	67,50 €	1	67,50 €
Processors	1012	AMD RYZEN 7 1700- Processor de 3.7 (	199,90 €	2	399,80 €
Motherboards	2003	Dell motherboard for OptiPlex GX170L 1	83,50 €	2	167,00 €
				Total:	869,17 €

Do the relational model in 3NF.

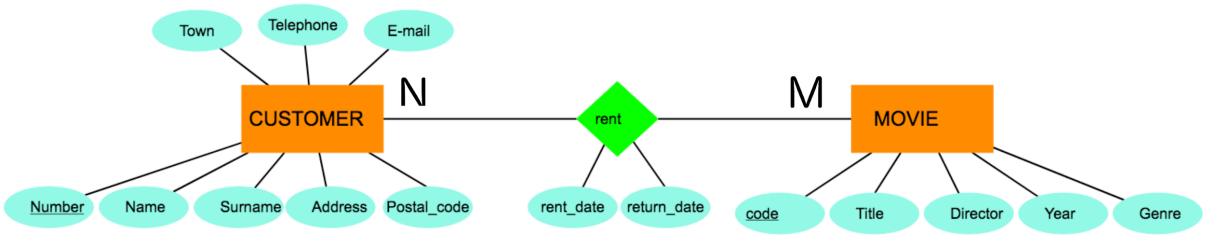
10.- Normalize up to 3NF the relation R (A, B, C, D, E, F) in which the following functional dependencies are given:

- E → B
- C → A
- C, E ⇒ F
- A → D

11.- Normalize up to 3NF the relation R (A, B, C, D, E) in which the following functional dependencies are given:

- C, E ⇒ D
- D → B
- C → A

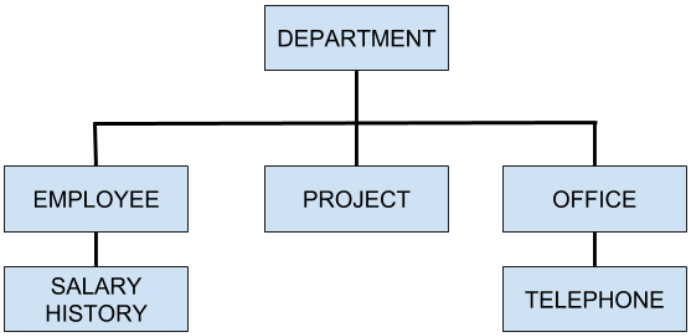
12.- Do the relational model in 3NF.
Entity-relationship model:



13.- Normalize up to 3NF.
R(Course, Teacher, Hour, Classroom, Student, Mark)
In which:

- Each course is done by a teacher.
- At one hour, in a classroom, a single course is done.
- At one hour, a teacher is only in a single classroom.
- Each student has a single mark for each course that he/she is enrolled.
- At one hour, a student is only in a single classroom.

14.- Normalize up to 3NF.



It is interesting to keep the following information:

- For each department: department number (unique), budget, and the number of the employee who is the department manager (unique).
- For each employee: employee number (unique), current project number, office number, personal telephone number, name of each job that has been held, description of each job, and amount and range of dates of each salary which he has received for each of these works.
- For each project: project number (unique) and budget.
- For each office: office number (unique), area in m^2, and numbers (unique) of all the phones in this office.

15.- Indicate the normal form of the following relationships and normalize them if necessary:
TRIPS (id_trip, departure_time, date, vehicle_plate, driver_id_card, driver_surname)
With the next functional dependencies:

- FD1: {id_trip, departure_time} → {vehicle_plate} → {driver_surname}
- FD1: {driver_id_card} → {driver_surname}

16.- An enterprise works with the following sheet:

Customer id	Name	Surname	Phone	Address
1	Brad	Pitt	90000001 60012312 61132199	504 Avenue, 23
2	Jennifer	Lawrence	97100012 97100013	525 Avenue, 3
3	George	Clooney	60012345	104 Avenue, 53
4	Jane	Fonda	97111111 97122222	14 Avenue, 2
...				

Customers (Customer id, Name, Surname, {Phone}, Address)

Translate to the 1NF, 2NF, and, finally, to the 3NF.

17.- An enterprise works with the following sheet:

Customer id	Name	Surname	Project	Budget	In charge	State
1	Brad	Pitt	Website12	12500	SGR	1
2	Jane	Fonda	DigiATM	50000	DGF	3
3	George	Clooney		49000		
1	Brad	Pitt	App15	25000	SGR	5

As you can see, a project can have many customers and a customer can have many projects. Every project has a single person in charge (and obviously that employee may be in charge of many projects).

CustomerProjects (Customer id, Name, Surname, Project, Budget, In charge, State)

Translate to the 1NF, 2NF, and, finally, to the 3NF.

18.- A shop rents video games. They work with a sheet with the following fields:

RENT (Videogame id, Title, Copy number, Customer num, Customer name, Customer phone number, rent date, rent number days)

Translate to the 1NF, 2NF, and, finally, to the 3NF.

19.- A college keeps data from students, professors, studies and subjects (inside those studies). The following considerations are assumed:

- A student must be enrolled in a single career (and maybe he/she is not enrolled in all the subjects of that career).
- There are subjects shared in different studies.
- A professor teaches a concrete subject (and he/she can teach many subjects).

They work with a sheet like this ones:

Student Id	Student Name	Student Surname	Subject Code	Subject Name	Professor Id	Professor Name	Studies Id	Studies Name
1001	Sergio	González	201	Databases	23	Michael Mann	2	Computer engineering
1001	Sergio	González	98	Programming I	12	Jane Smith	2	Computer engineering
1002	Laura	Palmer	98	Programming I	12	Jane Smith	1	Mathematics
...								

Translate to the 1NF, 2NF, and, finally, to the 3NF.

20.- A library is working with the following sheets:

Book Id	Title	Author	Editorial Id	Editorial Name	Reader Id	Reader Name	Start Date	End Date
12	Programming in C++	John Smith	12	Ra-Ma	23	Sergi González	12/1/17	19/1/17
23	Databases	Helen Furlong	14	McGraw Hill	28	Ana Ramis	15/2/18	22/2/18
12	Programming in C++	John Smith	12	Ra-Ma	23	Sergi González	1/3/17	8/3/17
...								

Translate to the 1NF, 2NF, and, finally, to the 3NF.

Course: Database management
Unit: Database Design. Normalization
Material: Normalization exercises solution
Teamwork: Groups of two or three

Here you will find a solution of one of the exercises. Evaluate the work of you classmates and let them evaluate your work.

1NF

The original table is not in the 1NF because it has multi-value attributes. We modify it as follows to get rid of the multi-valued attributes:

SID	CID	S_NAME	C_NAME	GRADE	FACULTY	F_PHONE
1	IS318	Adams	Database	A	Howser	60192
1	IS301	Adams	EC	B	Langley	45869
2	IS318	Jones	Database	A	Howser	60192
3	IS318	Smith	Database	B	Howser	60192
4	IS301	Baker	EC	A	Langley	45869
4	IS318	Baker	Database	B	Howser	60192

Students(SID, CID, S_NAME, C_NAME, GRADE, FACULTY, F_PHONE)

2NF

The database we obtained in the previous step is not in 2NF because we found some columns that depend functionally on only part of the primary key.
SID->SNAME
CID->CNAME
CID->FACULTY
CID->F_PHONE

To address this situation, we split the table in three different tables as follows.

Student(SID, S_NAME)

Takes(SID*, CID*, GRADE)
Course(CID, C_NAME, FACULTY, F_PHONE)

Student

SID	S_NAME
1	Adams
2	Jones
3	Smith
4	Baker

Takes

SID*	CID*	GRADE
1	IS318	A
1	IS301	B
2	IS318	A
3	IS318	B
4	IS301	A
4	IS318	B

Course

CID	C_NAME	FACULTY	F_PHONE
IS301	EC	Langley	45869
IS318	Database	Howser	60192

3NF

The database we obtained in the previous step is not in 3NF because there is a column that functionally depends on another column that is not part of the PK.

FACULTY->F_PHONE

To address this issue, we split the table Course in two different tables as follows

Student(SID, S_NAME)
Takes(SID*, CID*, GRADE)
Course(CID, C_NAME, FACULTY*)
Faculty(FACULTY, F_PHONE)

Student

SID	S_NAME
1	Adams
2	Jones
3	Smith
4	Baker

Takes

SID*	CID*	GRADE
1	IS318	A
1	IS301	B
2	IS318	A
3	IS318	B
4	IS301	A
4	IS318	B

Course

CID	C_NAME	FACULTY*
IS301	EC	Langley
IS318	Database	Howser

Faculty

FACULTY	F_PHONE
Langley	45869
Howser	60192

These tables are already in BCNF because there are no alternative candidate keys and all the attributes that are not part of the primary key depend fully functionally on it.

Course: Database management
Unit: Database Design. Normalization
Assignment: Normalization
Teamwork: Groups of two or three

Create a normalization problem and provide the solution. The solution should identify all functional dependencies and go through all the steps of the normalization process. At least, up to the 3NF.

Course: Database management
Unit: Database Design. Normalization
Assignment: Present Your Normalization
Teamwork: Groups of two or three

Present your normalization exercises and solutions and answer the questions of your classmates and the teachers..

Course: Database management
Unit: Database Design. Normalization
Assignment: Normalization Script
Teamwork: Groups of two or three

Install a RDBMS.

Create a Database that is not normalized.

Insert Data.

Create a script that normalizes the database keeping the data.

Deliver an explanatory report.

12 Unit 12: Advanced Topics

Course: Database management
Unit: Advanced topics.
Assignment: Advanced topics.
Teamwork: Groups of two or three

1. Create a database in a DBMS of your choice. Create a backup using the command line. Drop the database. Restore the backup and verify the result. Repeat this process using a graphical user interface.
2. Export the data of a table to a CSV file. Delete the data. Import the data from the file.
3. Find an error message of your DBMS. Explain what it means. Find the logs of your DBMS and explain part of its content.
4. Export data (or a full database) from a DBMS of your choice and move it to another DBMS. Document any difficulties or problems you found.

Course: Database management
Unit: Advanced topics
Material: NoSQL
Teamwork: Groups of two or three

In this work unit we will look at an example of a NoSQL (not only SQL) database which is MongoDB.

I recommend this quick getting started tutorial:

https://docs.mongodb.com/manual/tutorial/getting-started/

And also the M001 course of the MongoDB University:

https://university.mongodb.com/courses/M001/about

13 Google Classroom

All topics

Create

General Information

Class Diary	Posted Sep 23, 2019
Course Program	Posted Sep 23, 2019
grade proposal 1st term asix dual	Edited Mar 6, 2020
Provisional Grades	Posted Jun 1, 2020

0.- COVID-19 Task Force

Material 0-4 Partial data about the d...	Posted Mar 24, 2020
Material 0-1 Symptom Tracking DB 3	Posted Mar 18, 2020
Material 0-2 Multiperson Symptom T...	Posted Mar 20, 2020
Material 0-3 Video regarding contact...	Posted Mar 23, 2020

1.- Introduction to Databases

Example of classwork	Edited Oct 1, 2019
Test Introduction to Databases	Due Oct 2, 2019

2.- Introduction to the Relational Model

Introduction to the Relational Model	Posted Oct 1, 2019
Installing a Relational Database	Due Oct 7, 2019
Assignment_2-2_Keys	Due Oct 7, 2019
Example of Classwork "Installing a D...	Posted Oct 9, 2019
Assignment_2-4_Relational_Algebra	Due Oct 14, 2019
Grading Assignment 2-4 Relational A...	Posted Oct 15, 2019
Test_2_Introduction_to_the_Relation...	Due Oct 15, 2019
Anwers exam lesson 2 Introduction t...	Posted Oct 17, 2019
Grading Exam Lesson 2 Introduction ...	Posted Oct 17, 2019
Relational Diagram	Due Oct 22, 2019

Assignment_3-1_create_a_database	Due Oct 25, 2019
Example of classwork: Assignment_3...	Posted Oct 28, 2019
Assignment_3-2_ Dockerized_Datab...	Due Oct 31, 2019
Assignment_3-3_School_Database_S...	Due Nov 7, 2019
University script	Posted Nov 11, 2019
Link to download the university data...	Posted Nov 14, 2019
Requirements for the test	Posted Nov 18, 2019
Test lesson 3. Introduction to SQL 1	Due Nov 18, 2019, 6:20 ...
Lesson 3 Test results	Posted Nov 19, 2019

4.- Introduction to SQL 2

Practice material for queries	Posted Nov 19, 2019
Practice material for queries 2	Posted Nov 25, 2019
Northwind	Posted Dec 5, 2019
Test Introduction to SQL 2	Due Dec 17, 2019

Dockerized phpMyAdmin	Due Jan 13, 2020
Join exercises	Edited Jan 13, 2020
Views exercises	Posted Jan 17, 2020
Transactions	Posted Jan 24, 2020
Transactions02	Posted Jan 27, 2020
Test unit5 Intermediate SQL 1	Due Jan 30, 2020, 8:30 …
provisional results test 5	Posted Feb 3, 2020

6.- Intermediate SQL 2

Referential integrity and data types	Posted Jan 31, 2020
Mini League	Due Feb 14, 2020
Index importance	Posted Feb 13, 2020
Authorization	Posted Feb 14, 2020
Roles	Posted Feb 17, 2020
Dates	Edited Feb 20, 2020

Results	Posted Mar 3, 2020

7.- Advanced SQL 1

Connect to a database with java	Posted Feb 24, 2020
JDBC project and presentation	Due Mar 16, 2020
Assignment 7-2 Video Explanation an...	Due Mar 17, 2020

8.- Advanced SQL 2

Material 8-1 postgreSQL	Edited Sep 24, 2020
Material 8-2 PostgreSQL Stored Proc...	Posted Mar 20, 2020
Material 8-3 PostgreSQL Stored Proc...	Posted Mar 29, 2020
Material 8-4 PostgreSQL Stored Proc...	Posted Mar 30, 2020
Assignment 8-1 Create a Function	Due Apr 6, 2020
Assignment 8-2 function defense	Due Apr 15, 2020
Assignment 8-3 Create a cursor, a st...	Due Apr 24, 2020
Assignment 8-4 Defend your cursor, ...	Due Apr 28, 2020

Material 9-3 exercises (bis)	Posted May 6, 2020
Material 9-3 E-R Exercises	Posted May 5, 2020
Material 9-1 Slides	Posted Apr 23, 2020
Material 9-2 E-R Diagram Tool	Posted Apr 28, 2020
Assignment 9-1 Create an ERD	Due May 8, 2020
Assignment 9-2 Defend your ERD	Due May 13, 2020

10.- Database Design. ERM to Relational Model

10-1 ERM to Relational Model	Due May 19, 2020
Assignment 10-2 Present your Relati...	Due May 21, 2020

11.- Database Design. Normalization.

Material 11-1 Slides	Edited May 25, 2020
Material 11-2 Slides	Posted May 18, 2020
Material 11-3 Exercises 4	Posted May 18, 2020

Assignment 11-2 Present your Norma... Due Jun 1, 2020, 11:59 ...

12.- NoSQL

Material 12-1 NoSQL Edited Jun 1, 2020