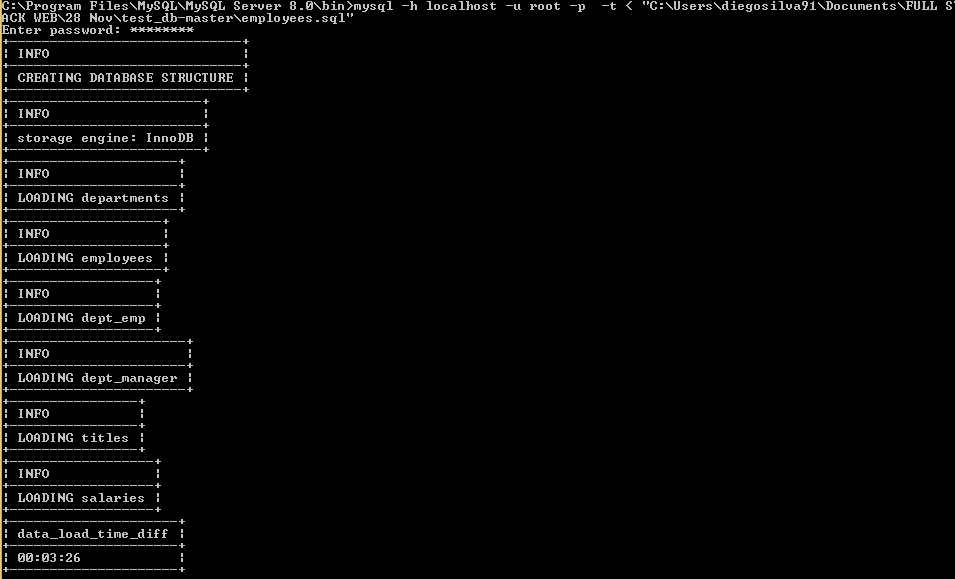
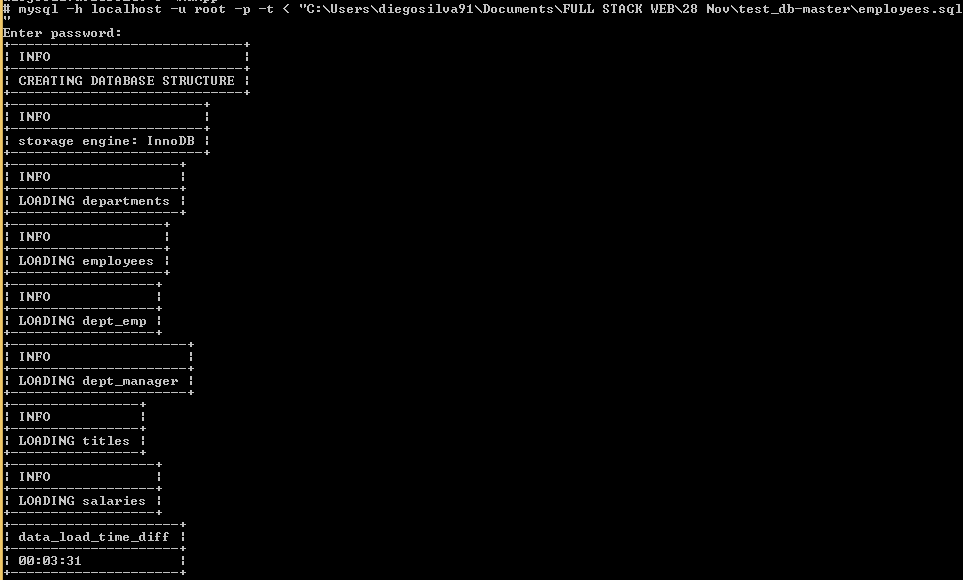
MYSQL BASICS

1. Install MySQL Server
2. Run the local server.
3. Import a sample data.





VALIDATING THE EMPLOYEE DATA



mysql -h localhost -u root -p -t < "C:\Users\diegosilva91\Documents\FULL STACK WEB\28 Nov\test\_db-master\test\_employees\_sha.sql"



1. Execute the following SQL queries.

show databases;

use employees;

* 1. INSERT DATA
     1. With salaries that are between a range of 5,000 and 50,000.

SELECT \* FROM salaries;

INSERT INTO salaries VALUES (0, 5442, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (1, 6474, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (2, 6493, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (3, 12328, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (4, 30450, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (5, 23334, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (6, 35341, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (7, 27334, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (8, 39341, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (9, 49334, '2015-09-23', '2015-10-23');

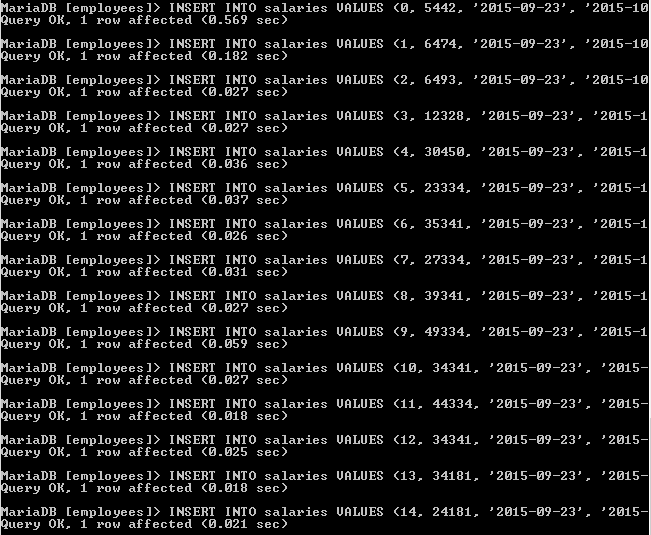
INSERT INTO salaries VALUES (10, 34341, '2015-09-23', '2015-10-23');

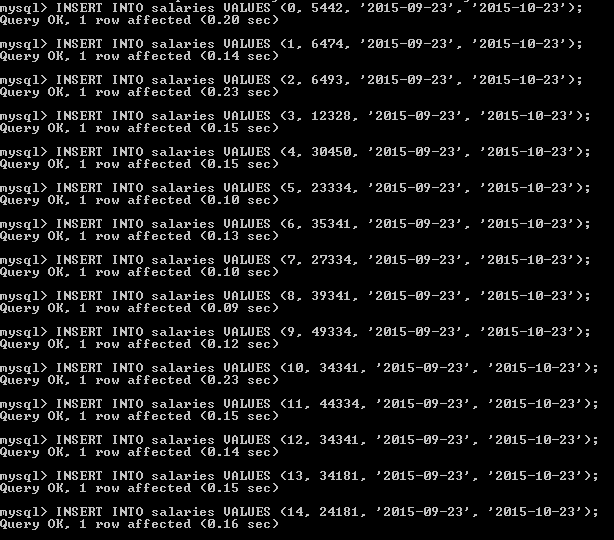
INSERT INTO salaries VALUES (11, 44334, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (12, 34341, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (13, 34181, '2015-09-23', '2015-10-23');

INSERT INTO salaries VALUES (14, 24181, '2015-09-23', '2015-10-23');





* + 1. Of different gender

show tables;

SELECT \* FROM employees;

INSERT INTO employees VALUES (0, '1999-09-20', 'Worker1', 'Work1 LastName', 'M', '2015-09-23');

INSERT INTO employees VALUES (1, '1999-09-20', 'Worker2', 'Work2 LastName', 'F', '2015-09-23');

INSERT INTO employees VALUES (2, '1999-09-20', 'Worker3', 'Work3 LastName', 'M', '2015-09-23');

INSERT INTO employees VALUES (3, '1999-09-20', 'Worker4', 'Work4 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (4, '1999-09-20', 'Worker5', 'Work5 Lastname', 'M', '2015-09-23');

INSERT INTO employees VALUES (5, '1999-09-20', 'Worker6', 'Work6 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (6, '1999-09-20', 'Worker7', 'Work7 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (7, '1999-09-20', 'Worker8', 'Work8 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (8, '1999-09-20', 'Worker9', 'Work9 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (9, '1999-09-20', 'Worker10', 'Work10 Lastname', 'M', '2015-09-23');

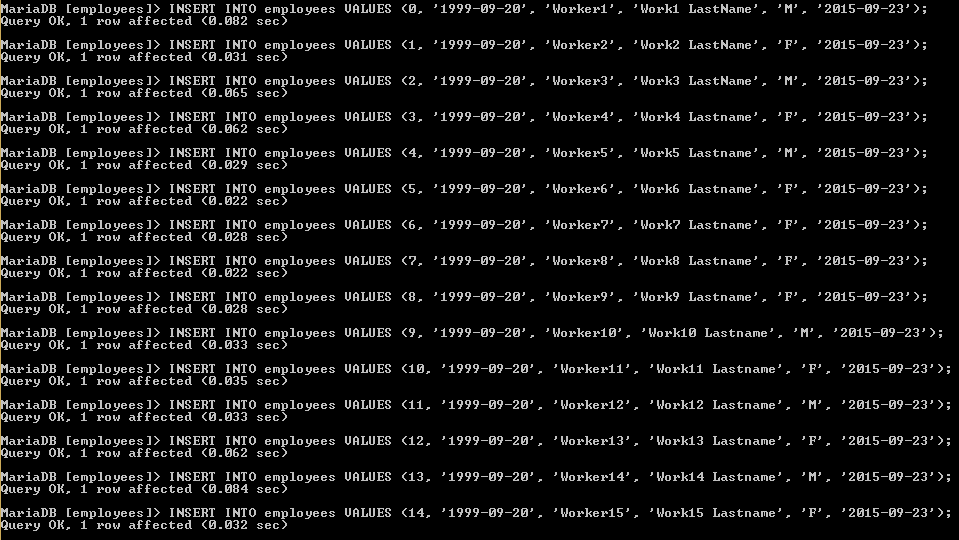
INSERT INTO employees VALUES (10, '1999-09-20', 'Worker11', 'Work11 Lastname', 'F', '2015-09-23');

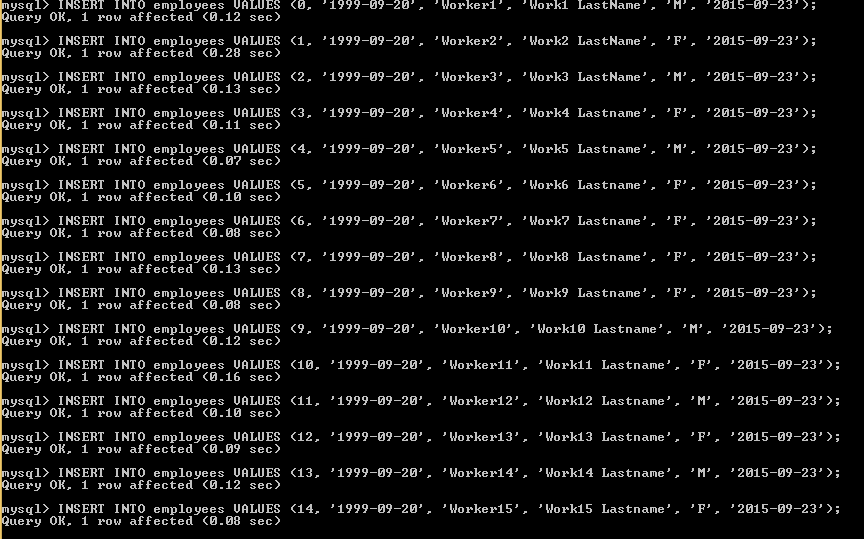
INSERT INTO employees VALUES (11, '1999-09-20', 'Worker12', 'Work12 Lastname', 'M', '2015-09-23');

INSERT INTO employees VALUES (12, '1999-09-20', 'Worker13', 'Work13 Lastname', 'F', '2015-09-23');

INSERT INTO employees VALUES (13, '1999-09-20', 'Worker14', 'Work14 Lastname', 'M', '2015-09-23');

INSERT INTO employees VALUES (14, '1999-09-20', 'Worker15', 'Work15 Lastname', 'F', '2015-09-23');





* + 1. 5 employees must have at least two salaries in different ranges of dates and different amount

INSERT INTO salaries VALUES (0, 6341, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (1, 10371, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (2, 4494, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (3, 13328, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (4, 34456, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (5, 33398, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (6, 33781, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (7, 24566, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (8, 44768, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (9, 23455, '2015-10-23', '2015-11-23');

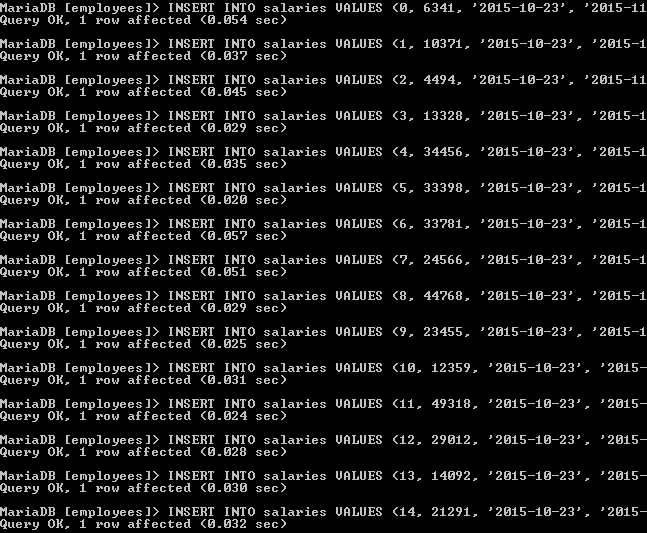
INSERT INTO salaries VALUES (10, 12359, '2015-10-23', '2015-11-23');

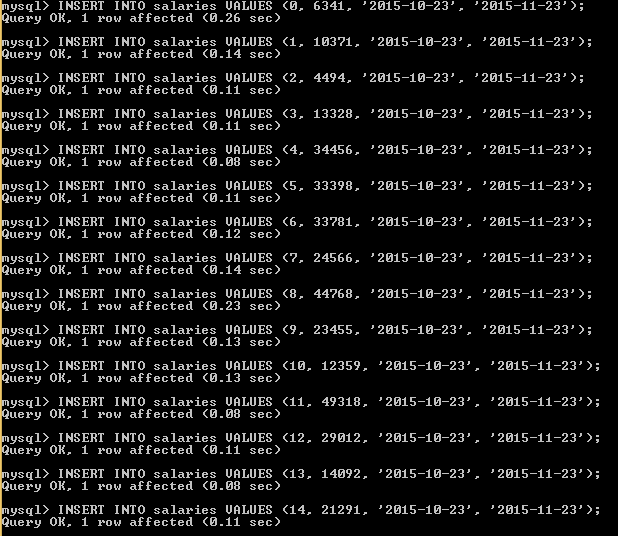
INSERT INTO salaries VALUES (11, 49318, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (12, 29012, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (13, 14092, '2015-10-23', '2015-11-23');

INSERT INTO salaries VALUES (14, 21291, '2015-10-23', '2015-11-23');





* + 1. 10 employees belong to more than one department.

INSERT INTO dept\_emp VALUES (0, 'd004', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (0, 'd006', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (1, 'd001', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (2, 'd002', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (3, 'd003', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (4, 'd004', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (5, 'd005', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (5, 'd008', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (6, 'd006', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (6, 'd008', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (7, 'd007', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (7, 'd009', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (8, 'd008', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (8, 'd004', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (9, 'd009', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (9, 'd002', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (10, 'd004', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (10, 'd008', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (11, 'd001', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (11, 'd006', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (12, 'd002', '2015-09-23', '2017-10-23');

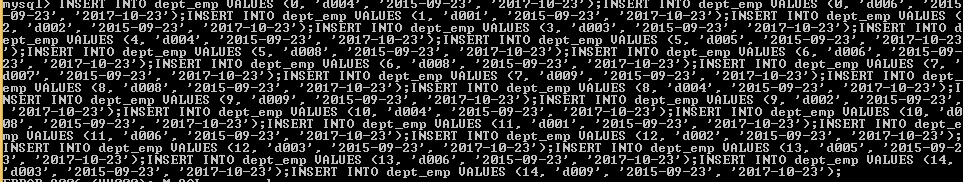
INSERT INTO dept\_emp VALUES (12, 'd003', '2015-09-23', '2017-10-23');

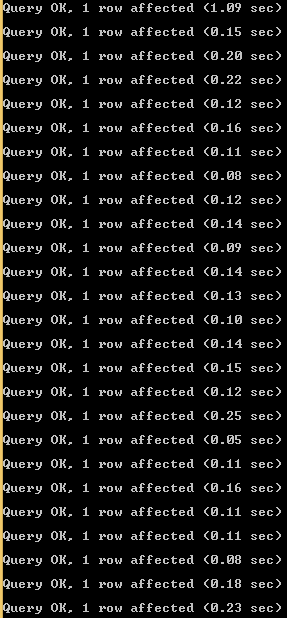
INSERT INTO dept\_emp VALUES (13, 'd005', '2015-09-23', '2017-10-23');

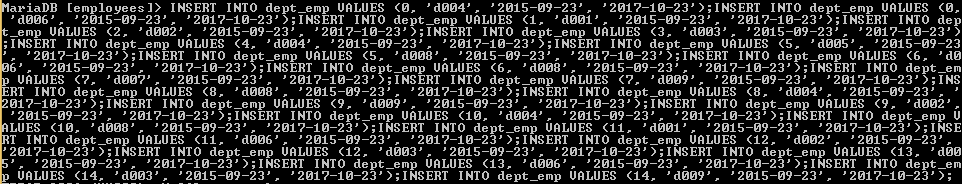
INSERT INTO dept\_emp VALUES (13, 'd006', '2015-09-23', '2017-10-23');

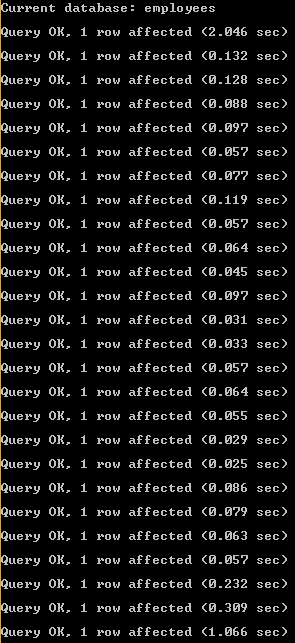
INSERT INTO dept\_emp VALUES (14, 'd003', '2015-09-23', '2017-10-23');

INSERT INTO dept\_emp VALUES (14, 'd009', '2015-09-23', '2017-10-23');









* + 1. 5 employees are managers.

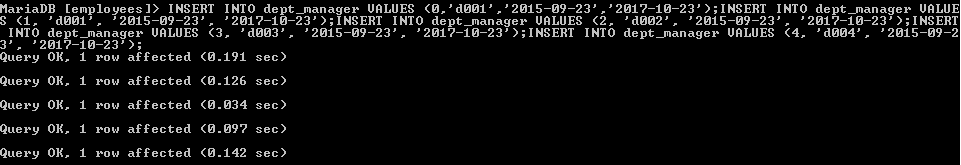
INSERT INTO dept\_manager VALUES (0,'d001','2015-09-23','2017-10-23');

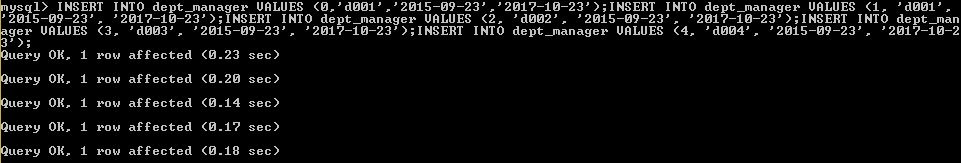
INSERT INTO dept\_manager VALUES (1, 'd001', '2015-09-23', '2017-10-23');

INSERT INTO dept\_manager VALUES (2, 'd002', '2015-09-23', '2017-10-23');

INSERT INTO dept\_manager VALUES (3, 'd003', '2015-09-23', '2017-10-23');

INSERT INTO dept\_manager VALUES (4, 'd004', '2015-09-23', '2017-10-23');





* + 1. All employees have a degree and at least 5 titles are from 2019.

INSERT INTO titles VALUES (0, 'Technique Leader', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (1, 'Staff', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (2, 'Engineer', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (3, 'Senior Engineer', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (4, 'Assistant Engineer', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (5, 'Technique Leader', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (6, 'Senior Staff', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (7, 'Senior Staff', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (8, 'Engineer', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (9, 'Technique Leader', '2015-09-23', '2017-10-23');

INSERT INTO titles VALUES (10, 'Technique Leader', '2015-09-23', '2019-10-23');

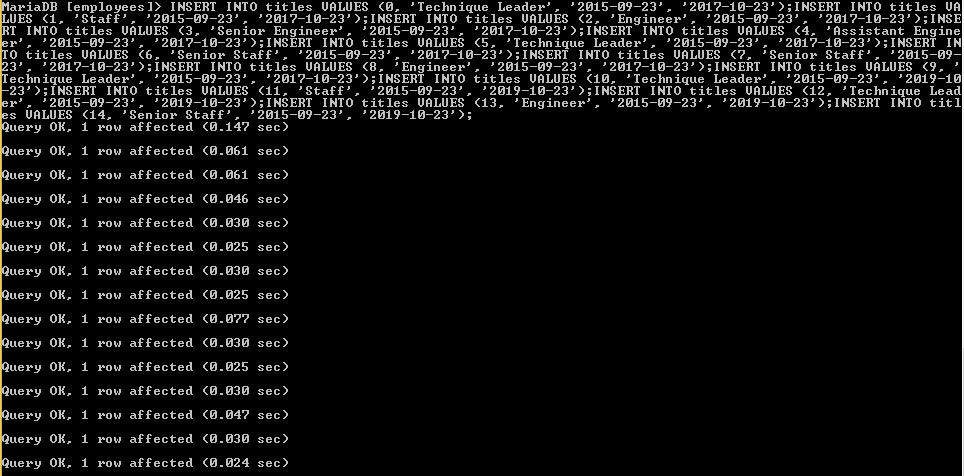
INSERT INTO titles VALUES (11, 'Staff', '2015-09-23', '2019-10-23');

INSERT INTO titles VALUES (12, 'Technique Leader', '2015-09-23', '2019-10-23');

INSERT INTO titles VALUES (13, 'Engineer', '2015-09-23', '2019-10-23');

INSERT INTO titles VALUES (14, 'Senior Staff', '2015-09-23', '2019-10-23');





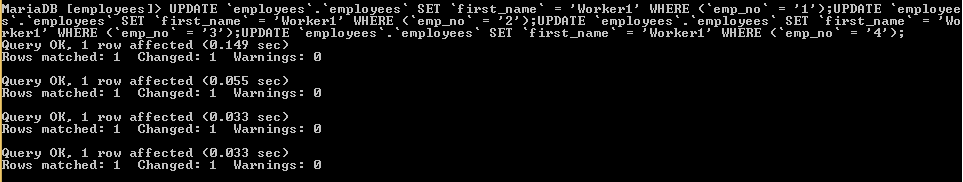
* + 1. At least 3 employees have the same name

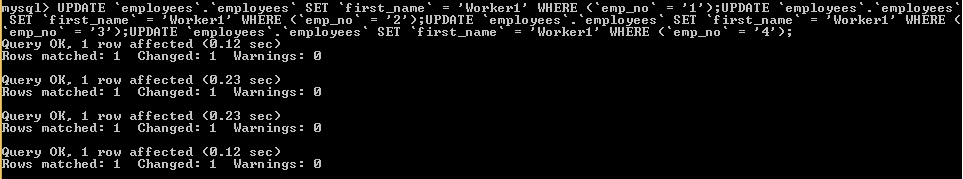
UPDATE `employees`.`employees` SET `first\_name` = 'Worker1' WHERE (`emp\_no` = '1');

UPDATE `employees`.`employees` SET `first\_name` = 'Worker1' WHERE (`emp\_no` = '2');

UPDATE `employees`.`employees` SET `first\_name` = 'Worker1' WHERE (`emp\_no` = '3');

UPDATE `employees`.`employees` SET `first\_name` = 'Worker1' WHERE (`emp\_no` = '4');





* 1. UPDATE DATA
* UPDATE EMPLOYEES

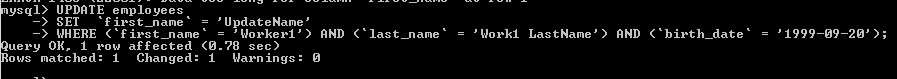
Change the name of an employee. To do this, generate a query that affects only a certain employee based on their name, surname and date of birth.

UPDATE employees

SET `first\_name` = 'UpdateName'

WHERE (`first\_name` = 'Worker1') AND (`last\_name` = 'Work1 LastName') AND (`birth\_date` = '1999-09-20');





* UPDATE DEPARTMENTS

UPDATE departments

SET `dept\_name` = 'Engineer'

WHERE (`dept\_no` = 'd001');

UPDATE departments

SET `dept\_name` = 'Desing'

WHERE (`dept\_no` = 'd002');

UPDATE departments

SET `dept\_name` = 'Marketing'

WHERE (`dept\_no` = 'd003');

UPDATE departments

SET `dept\_name` = 'HR'

WHERE (`dept\_no` = 'd004');

UPDATE departments

SET `dept\_name` = 'Sowftware'

WHERE (`dept\_no` = 'd005');

UPDATE departments

SET `dept\_name` = 'Quality'

WHERE (`dept\_no` = 'd006');

UPDATE departments

SET `dept\_name` = 'Suppliers'

WHERE (`dept\_no` = 'd007');

UPDATE departments

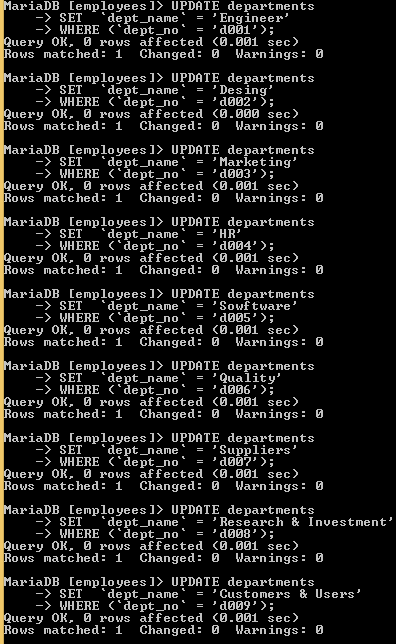
SET `dept\_name` = 'Research & Investment'

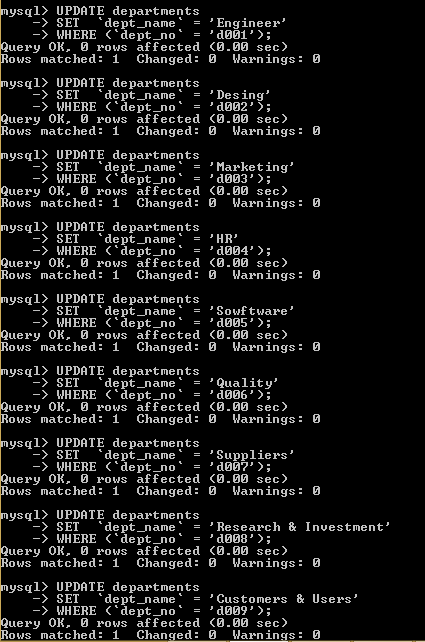
WHERE (`dept\_no` = 'd008');

UPDATE departments

SET `dept\_name` = 'Customers & Users'

WHERE (`dept\_no` = 'd009');

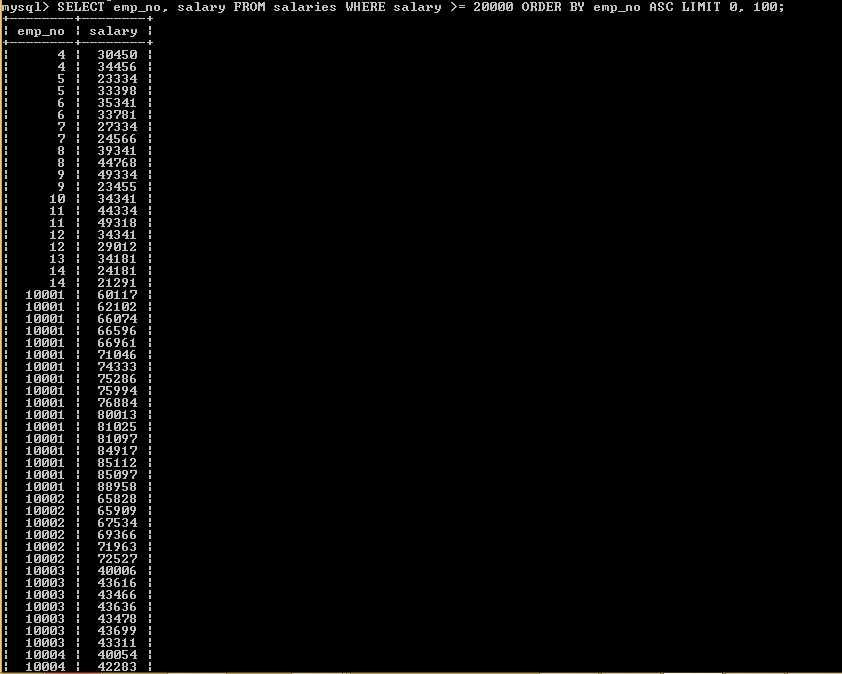




* 1. GET DATA
* Select all the employees with a salary greater than 20000.

To show only salaries and employee’s number from the table salaries:

SELECT emp\_no, salary FROM salaries WHERE salary >= 20000;



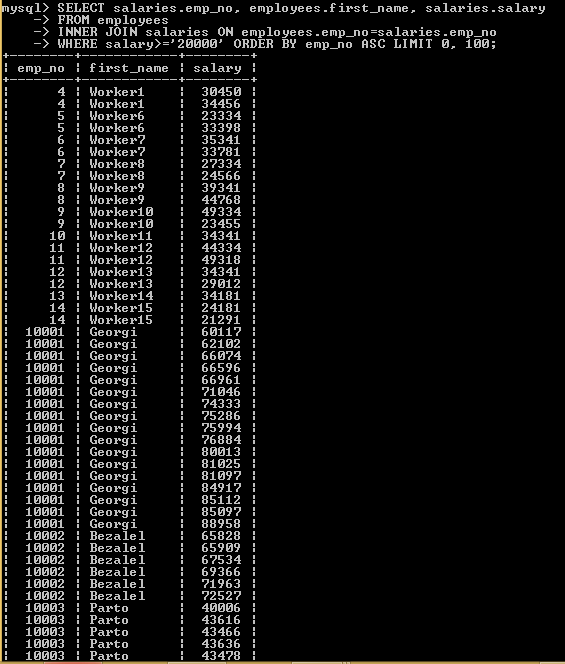
To show employee’s number, name and salaries from the table salaries and employees:

SELECT salaries.emp\_no, employees.first\_name, salaries.salary

FROM employees

INNER JOIN salaries ON employees.emp\_no=salaries.emp\_no

WHERE salary>='20000';



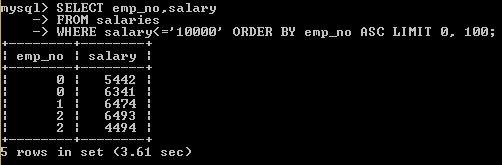
* Select all employees with a salary below 10,000.

To show only salaries and employee’s number from the table salaries:

SELECT emp\_no,salary

FROM salaries

WHERE salary<='10000';



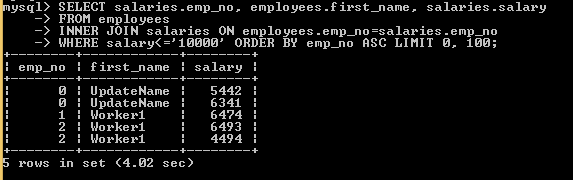
To show employee’s number, name and salaries from the table salaries and employees:

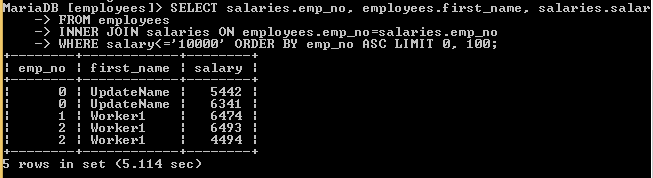
SELECT salaries.emp\_no, employees.first\_name, salaries.salary

FROM employees

INNER JOIN salaries ON employees.emp\_no=salaries.emp\_no

WHERE salary<='10000';



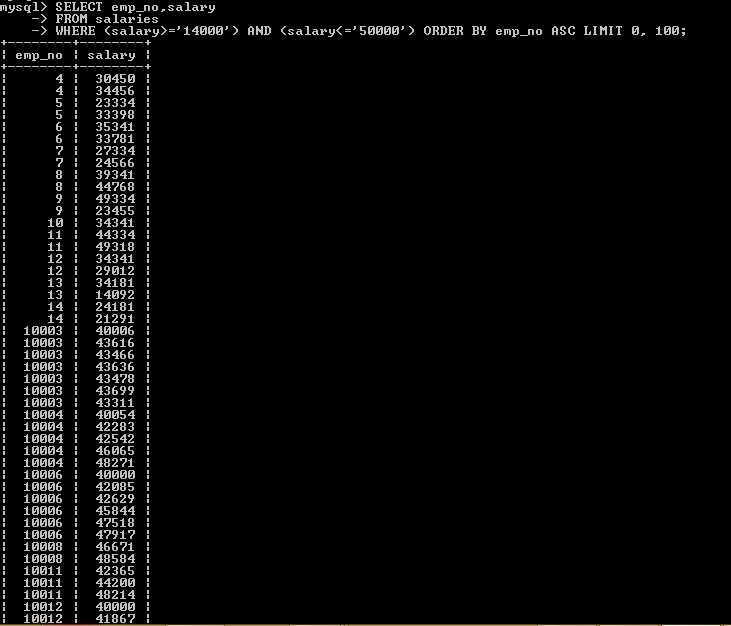


* Select all employees who have a salary between 14.00 and 50,000

SELECT emp\_no,salary

FROM salaries

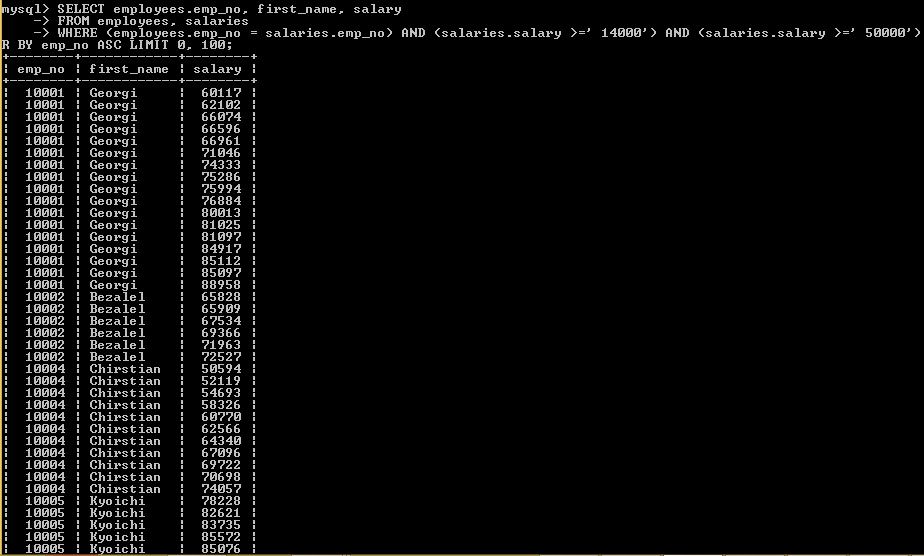
WHERE (salary>='14000') AND (salary<='50000');



SELECT employees.emp\_no, first\_name, salary

FROM employees, salaries

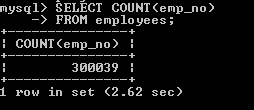
WHERE (employees.emp\_no = salaries.emp\_no) AND (salaries.salary >=' 14000') AND (salaries.salary >=' 50000');



* Select the total number of employees

SELECT COUNT(emp\_no)

FROM employees;



* Select the total number of employees who have worked in more than one department

SELECT emp\_no, COUNT(dept\_no)

FROM dept\_emp

GROUP BY emp\_no

HAVING COUNT(dept\_no) >=2;



* Select the titles of the year 2019

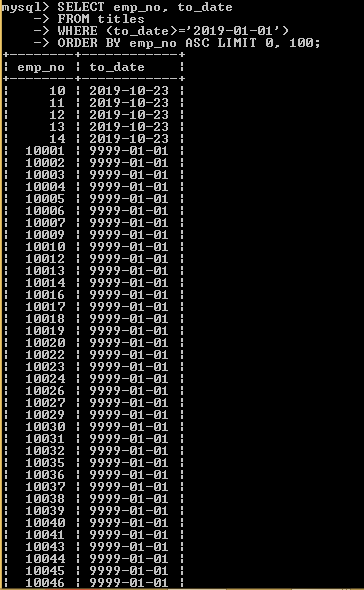
SELECT COUNT(emp\_no)

FROM employees;

SELECT emp\_no, to\_date

FROM titles

WHERE (to\_date>='2019-01-01');

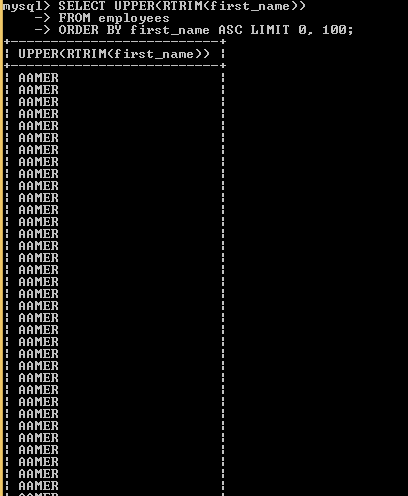


* Select only the name of the employees in capital letters

SELECT UPPER(RTRIM(first\_name))

FROM employees

ORDER BY first\_name;



* Select the name, surname and name of the current department of each employee

SELECT dept\_emp.emp\_no, employees.first\_name, employees.last\_name, MAX(departments.dept\_name) AS dept\_name, MAX(dept\_emp.from\_date) AS from\_date, MAX(dept\_emp.to\_date) AS to\_date

FROM dept\_emp

INNER JOIN departments ON dept\_emp.dept\_no=departments.dept\_no

INNER JOIN employees ON dept\_emp.emp\_no=employees.emp\_no

WHERE DATE(dept\_emp.to\_date)>= CURDATE() OR DATE(dept\_emp.from\_date)>= CURDATE()

GROUP BY emp\_no;



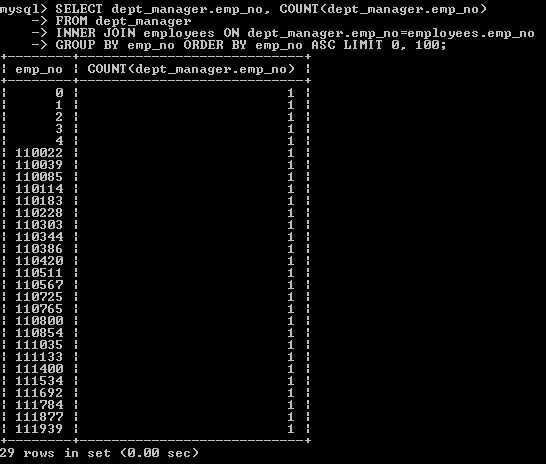
* Select the name, surname and number of times the employee has worked as a manager.

SELECT dept\_manager.emp\_no, COUNT(dept\_manager.emp\_no)

FROM dept\_manager

INNER JOIN employees ON dept\_manager.emp\_no=employees.emp\_no

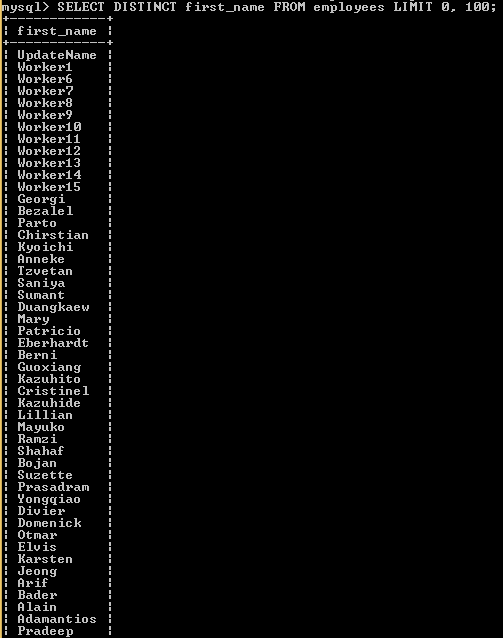
GROUP BY emp\_no;



* Select the name without any being repeated

SELECT DISTINCT first\_name

FROM employees;



* 1. DELETE DATA
* Eliminate all employees with a salary greater than 20,000.

This query insert the data of an employee who have salary greater than 20,000 and data of another tables. If the query to delete works as well, it have to delete all records of this employee too.

INSERT INTO employees VALUES (29, '1991-01-29', 'Diego', 'Silva', 'M', '2019-09-23');

INSERT INTO salaries VALUES (29, 1100000, '2019-09-23', '2019-10-23');

INSERT INTO salaries VALUES (29, 1100000, '2019-10-23', '2019-11-23');

INSERT INTO salaries VALUES (29, 1100000, '2019-11-23', '2019-11-23');

INSERT INTO dept\_emp VALUES (29, 'd004', '2019-09-23', '2019-12-23');

INSERT INTO dept\_emp VALUES (29, 'd005', '2019-09-23', '2019-12-23');

INSERT INTO dept\_manager VALUES (29,'d001','2019-10-23','2019-12-23');

INSERT INTO titles VALUES (29, 'Technique Leader', '2017-09-23', '2019-10-23');

Delete the data that satisfy the conditions

DELETE salaries, employees, dept\_emp, dept\_manager,titles FROM employees

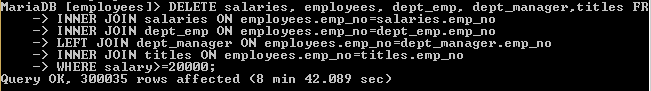
INNER JOIN salaries ON employees.emp\_no=salaries.emp\_no

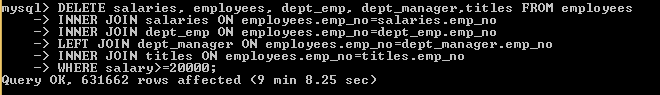
INNER JOIN dept\_emp ON employees.emp\_no=dept\_emp.emp\_no

LEFT JOIN dept\_manager ON employees.emp\_no=dept\_manager.emp\_no

INNER JOIN titles ON employees.emp\_no=titles.emp\_no

WHERE salary>=20000;





Then select the data after to delete, to check the delete’s query works as well.

SELECT salary, salaries.emp\_no, employees.first\_name, employees.last\_name, dept\_emp.dept\_no, titles.title, dept\_manager.dept\_no

FROM salaries

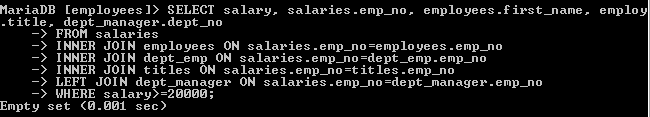
INNER JOIN employees ON salaries.emp\_no=employees.emp\_no

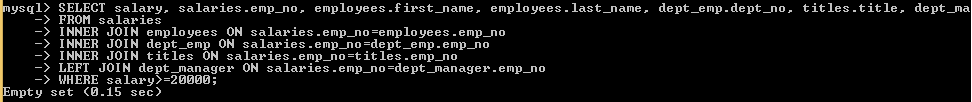
INNER JOIN dept\_emp ON salaries.emp\_no=dept\_emp.emp\_no

INNER JOIN titles ON salaries.emp\_no=titles.emp\_no

LEFT JOIN dept\_manager ON salaries.emp\_no=dept\_manager.emp\_no

WHERE salary>=20000;





* Remove the department that has more employees.

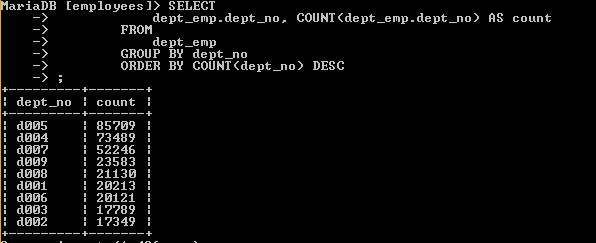
First check before to delete the data, the deparment that has more employees (it doesn’t necessary to check before delete)

SELECT dept\_no, COUNT(dept\_no) AS count

FROM dept\_emp

GROUP BY dept\_no

order by count DESC;



Then execute the query to delete

DELETE dept\_emp , employees, salaries, titles, dept\_manager FROM employees

INNER JOIN

dept\_emp ON employees.emp\_no = dept\_emp.emp\_no

INNER JOIN

salaries ON employees.emp\_no = salaries.emp\_no

INNER JOIN

titles ON employees.emp\_no = titles.emp\_no

left JOIN

dept\_manager ON employees.emp\_no = dept\_manager.emp\_no

WHERE

dept\_emp.dept\_no IN (SELECT

dept\_no

FROM

(SELECT

dept\_emp.dept\_no, COUNT(dept\_emp.dept\_no) AS count

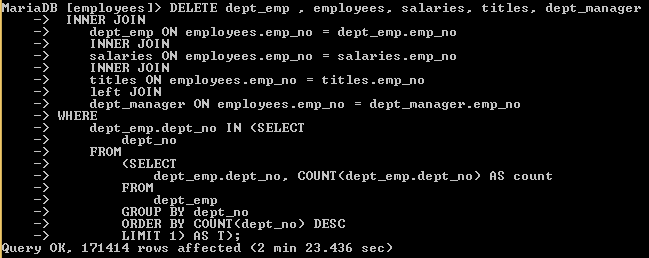
FROM

dept\_emp

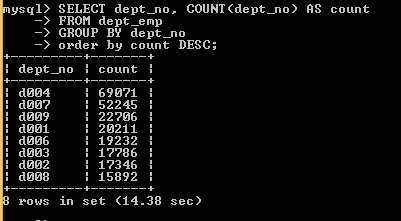
GROUP BY dept\_no

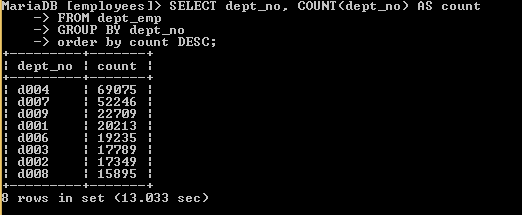
ORDER BY COUNT(dept\_no) DESC

LIMIT 1) AS T);



After delete the data, run the query again to check if the data has removed. The department that contains more employees has removed.





The results contains all departments without the first of the last group of results. To check in the main table “employees”, the department

SELECT employees.emp\_no, dept\_emp.dept\_no

FROM employees

INNER JOIN dept\_emp ON employees.emp\_no=dept\_emp.emp\_no

where EXISTS (SELECT dept\_emp.dept\_no

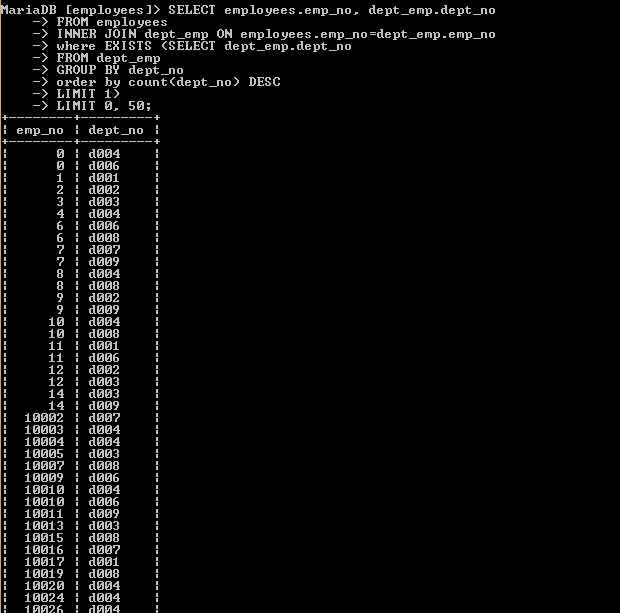
FROM dept\_emp

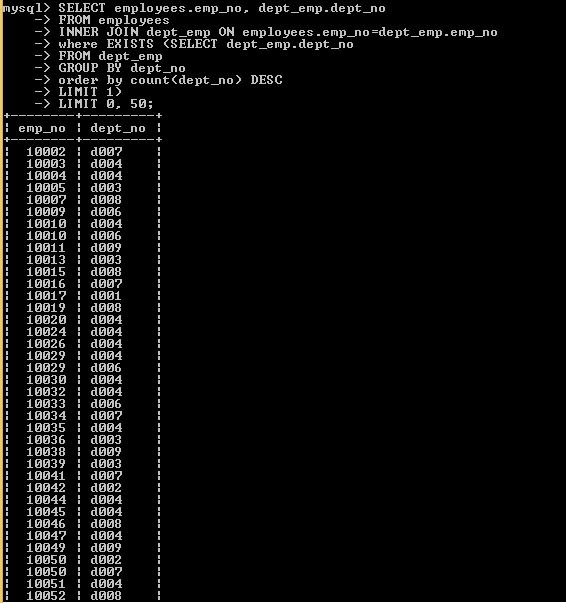
GROUP BY dept\_no

order by count(dept\_no) DESC

LIMIT 1)

LIMIT 0, 50;





1. CREATE A PRESENTATION

* How to import a database
* How to make inquiries
* What kind of queries exist and their syntax

In SQL we have quite a few sentences that can be used to perform various tasks.

Depending on the tasks, these sentences can be classified into three main groups (DML, DDL, DCL), although we would have another group that I think is not within the SQL language but the PLSQL.



* What kind of keys exist?

To ensure the integrity of the data stored in our tables, we can create restrictions, some of us have used them unintentionally or we simply do not know that what we did was a restriction, for example a primary key. These restrictions can be implemented at the time of creating our tables or modifying them, it is also necessary to point out that these restrictions are objects belonging to the database and therefore require a unique name composed of the name of the scheme to which it belongs and the name that identifies it, an example would be SchemaName.restrictionName.

The different types of restrictions that exist are:

* PRIMARY KEY.

It is the most common of all because each of our tables must be completely relational and to achieve this there must always be a primary key within each table that identifies each row as unique.

It is possible to add more columns as part of a primary key, it is recommended as good practice to use a nomenclature in the name of the restriction that helps identify what type it is, in addition to taking special care to name the columns that are part of the key primary as these are used as a reference in a foreign key in another table. Each time we generate a primary key, it creates an index type of clustered automatically.

There are certain requirements for the creation of a primary key:

* The column (s) used in a PRIMARY KEY constraint cannot accept NULL.
* You cannot repeat values ​​in the column (s), they must be unique.
* Only one PRIMARY KEY type constraint can exist for each table.
* UNIQUE.

This type of restriction is very similar to PRIMARY KEY, the differences are as follows:

* It also generates an index automatically but is of the NON CLUSTERED type.
* The table can have more than one UNIQUE type constraint.
* If you can accept NULL, but only one row can contain it since, as the name implies, it is UNIQUE or unique.
* FOREIGN KEY.

It is formed of a column or the combination of several columns of a table that serves as a link to another table where in the latter, said link the column is or columns that form the PRIMARY KEY. In the first table where we create the foreign key it is possible that there are duplicate values ​​of the column (s) that make up the primary key of the second table, in addition the columns involved in the foreign key must have the same type of data as the primary key of The second table. A foreign key does not create an index automatically, so it is recommended to generate one to increase the query performance.

Some requirements for the FOREIGN KEY restriction:

The values ​​entered in the foreign key column (s) must exist in the table referred to in the primary key column (s).

You can only refer to primary keys of tables that are within the same database.

You can refer to other columns in the same table.

You can only refer to PRIMARY KEY or UNIQUE constraint columns.

It cannot be used in temporary tables.

* CHECK.

With this type of restriction, it is specified that the values ​​entered in the column must comply with the specified rule or formula.

Some requirements are:

* A column can have any number of CHECK restrictions.
* The search condition must be evaluated as a Boolean expression and cannot refer to another table.
* You cannot define CHECK restrictions in columns of type text, ntext or image.

Advantage:

* The expressions used are similar to those used in the WHERE clause.
* They can become a better alternative than TRIGGERS or triggers.

Always keep in mind:

* When creating our expression, take into account if the column accepts NULL values, for example if we define our restriction that accepts only positive values ​​(columnName1> = 0), NULL is an unknown value therefore it will be inserted in the column.
* It is not possible to obtain the previous value after performing an UPDATE, if this is necessary it is recommended to use a TRIGGER.
* DEFAULT

It can be said that it is not a restriction, since only one value is entered in case no other is specified. If a column allows NULL and the value to be inserted is not specified, it can be replaced with a default value.

* What kind of data exists and what is each used for



* What is a collation, such as utf8\_general\_ci or utf8\_spanish\_ci and what are its differences

Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case-sensitivity, accent marks, kana character types and character width.

These two collations are both for the UTF-8 character encoding. The differences are in how text is sorted and compared.

* utf8mb4\_unicode\_ci is based on the official Unicode rules for universal sorting and comparison, which sorts accurately in a wide range of languages.
* Utf8mb4\_general\_ci is a simplified set of sorting rules which aims to do as well as it can while taking many short-cuts designed to improve speed. It does not follow the Unicode rules and will result in undesirable sorting or comparison in some situations, such as when using particular languages or characters.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT MAX(count) FROM(SELECT dept\_emp.dept\_no, COUNT(dept\_emp.dept\_no) AS count

FROM dept\_emp

INNER JOIN employees ON dept\_emp.emp\_no=employees.emp\_no

GROUP BY dept\_no) AS T;

SELECT dept\_no FROM(SELECT dept\_emp.dept\_no, COUNT(dept\_emp.dept\_no) AS count

FROM dept\_emp

GROUP BY dept\_no) AS T1

WHERE count IN (SELECT MAX(count)

FROM(SELECT dept\_emp.dept\_no, COUNT(dept\_emp.dept\_no) AS count

FROM dept\_emp

GROUP BY dept\_no) AS T2

);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*dept and count most employees\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

DELETE dept\_emp , employees FROM employees

INNER JOIN

dept\_emp ON employees.emp\_no = dept\_emp.emp\_no

WHERE dept\_no IN(SELECT dept\_no FROM(SELECT dept\_emp.dept\_no, COUNT(dept\_emp.dept\_no) AS count

FROM dept\_emp

GROUP BY dept\_no

order by count(dept\_no) DESC

LIMIT 1)AS T);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/