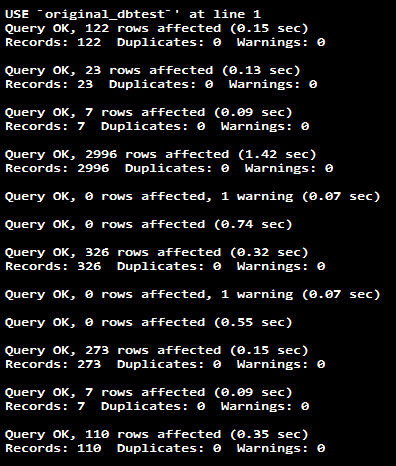
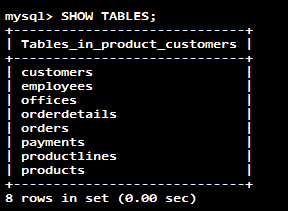
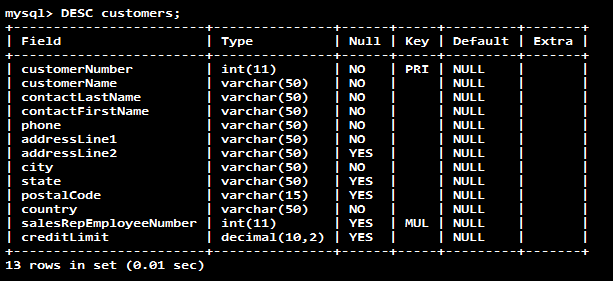
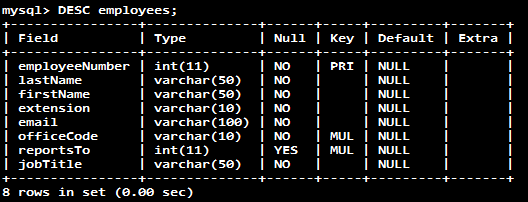
Lancer le script schema-data.sql

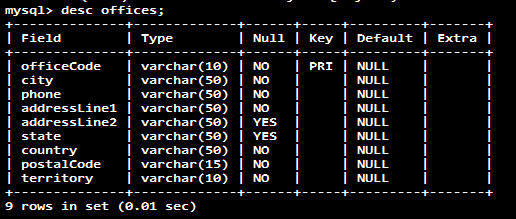
Recréer la DB en créant trois scripts : schema, data, constraint.sql

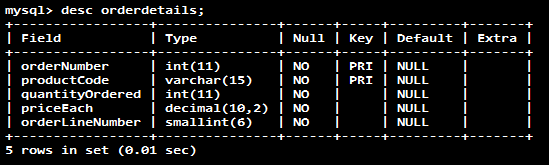


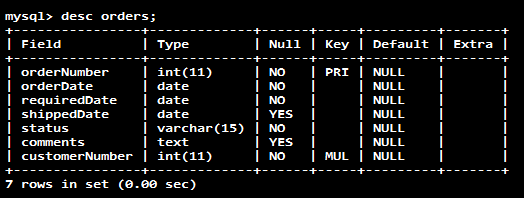


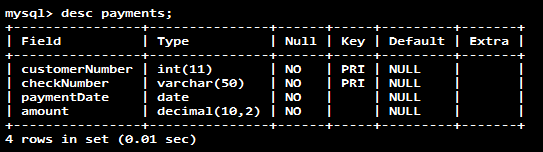


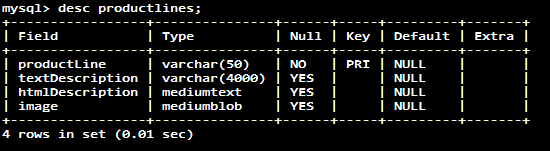


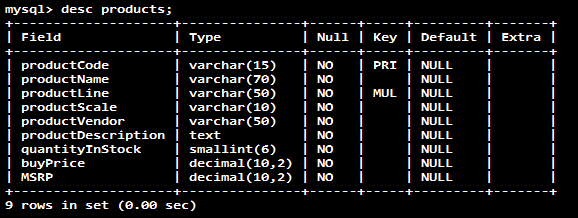












-- select complet

SELECT

column\_1, column\_2, ...

FROM

table\_1

[INNER | LEFT |RIGHT] JOIN table\_2 ON conditions

WHERE

conditions

GROUP BY column\_1

HAVING group\_conditions

ORDER BY column\_1

LIMIT offset, length;

-- Qry : SELECT

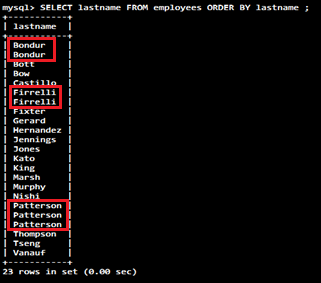
SELECT lastname, firstname, jobtitle FROM employees ORDER BY lastname ;



-- Qry : DISTINCT

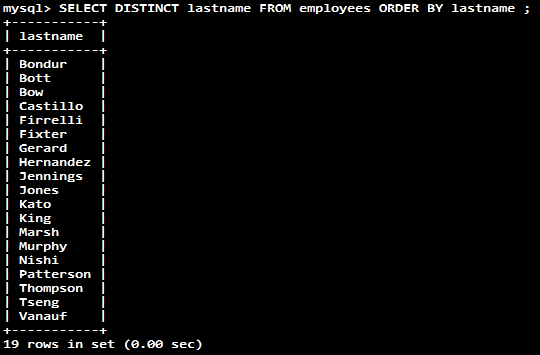
SELECT lastname FROM employees ORDER BY lastname ;

-- Rq : Certains employés ont leur même nom de famille : Bondur, Firrelli et Patterson !



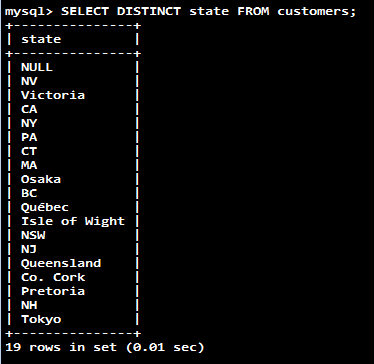
-- Q : Supprimer les doublons !

SELECT DISTINCT lastname FROM employees ORDER BY lastname ;



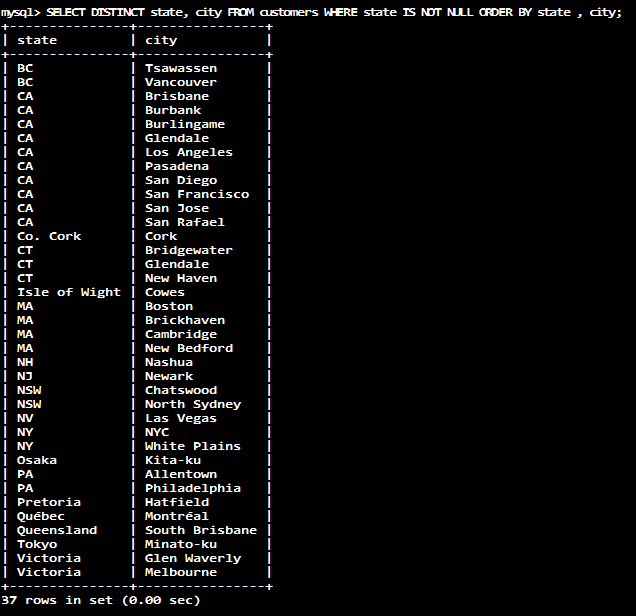
-- Qry : DISTINCT + NULL values

SELECT DISTINCT state FROM customers;



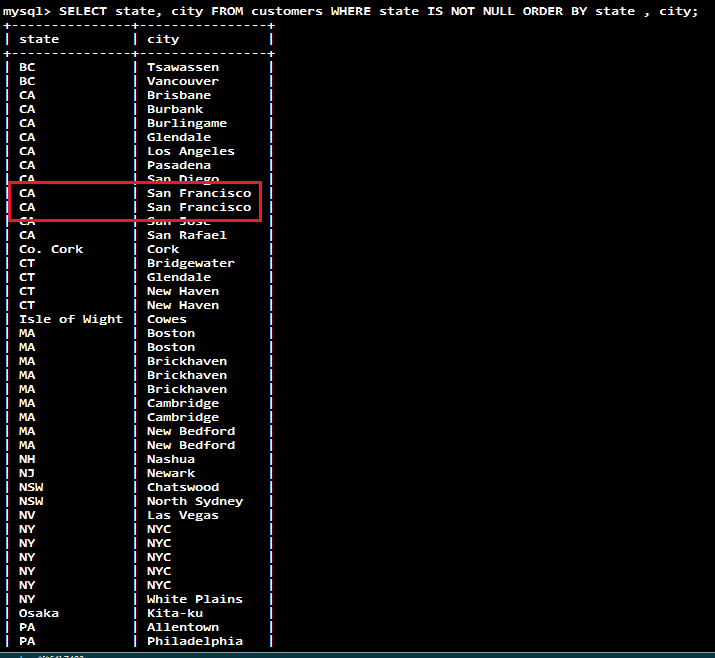
-- Qry : DISTINCT + Multiple Columns

SELECT DISTINCT state, city FROM customers WHERE state IS NOT NULL ORDER BY state , city;



SELECT state, city FROM customers WHERE state IS NOT NULL ORDER BY state , city;

-- CA SAN Francisco se repete 2 fois



-- Qry DISTINCT Vs. GROUP BY

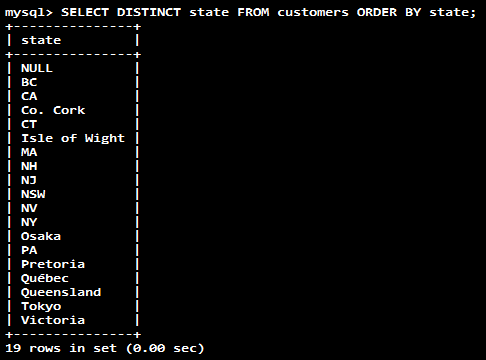
SELECT state FROM customers GROUP BY state;



-- GROUP BY retourne des données triés.

-- On n'a pas besoin d'1 ORDER BY avec GROUP BY (sans 1 fonction d'AGGREGATION)

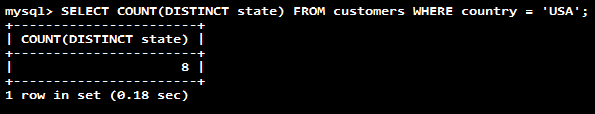
SELECT DISTINCT state FROM customers ORDER BY state;



-- DISTINCT ne trie pas les données. Ns sommes obligés d'indiquer ORDER BY.

-- DISTINCT avec 1 fonction d'aggrégation

SELECT COUNT(DISTINCT state) FROM customers WHERE country = 'USA';



-- Qry : ORDER BY + ALIAS

SELECT ordernumber, orderlinenumber,

quantityOrdered \* priceEach AS subtotal

FROM

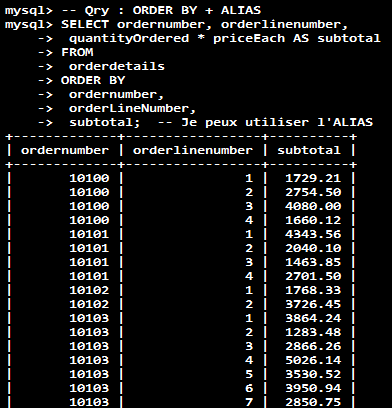
orderdetails

ORDER BY

ordernumber,

orderLineNumber,

subtotal; -- Je peux utiliser l'ALIAS



-- Qry : ORDER BY + FIELD

SELECT

orderNumber, status

FROM

orders

ORDER BY FIELD(status,

'In Process',

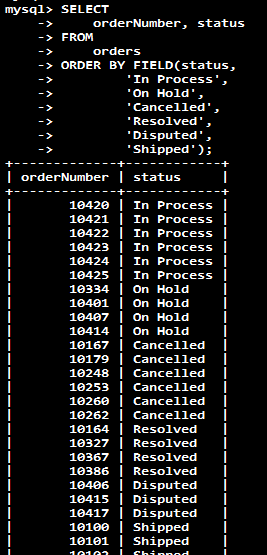
'On Hold',

'Cancelled',

'Resolved',

'Disputed',

'Shipped');



-- Qry : WHERE + FIELD

SELECT

lastname,

firstname,

jobtitle

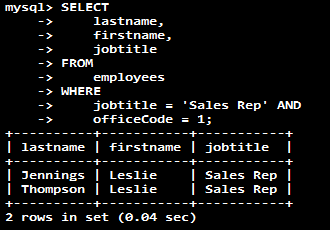
FROM

employees

WHERE

jobtitle = 'Sales Rep' AND

officeCode = 1;



-- Qry : INNER JOIN

SELECT productCode, productName, textDescription FROM

products t1

INNER JOIN

productlines t2 ON t1.productline = t2.productline;



-- Qry : INNER JOIN + GROUP BY

-- Q : Trouver le total des ventes par 'orderNumber' et 'status' ?

SELECT T1.orderNumber, status,

SUM(quantityOrdered \* priceEach) total

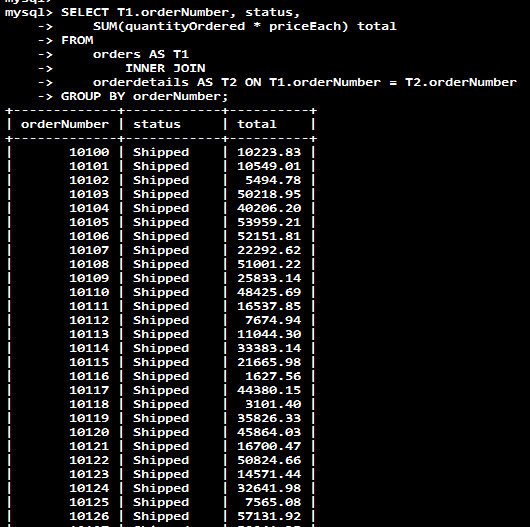
FROM

orders AS T1

INNER JOIN

orderdetails AS T2 ON T1.orderNumber = T2.orderNumber

GROUP BY orderNumber;



-- Qry : INNER JOIN + GROUP BY + WHERE

-- Q : find sales prices of the product whose code is S10\_1678 that are less than the manufacturer’s suggested retail price (MSRP) for that product ?

SELECT orderNumber, productName, msrp, priceEach FROM

products p

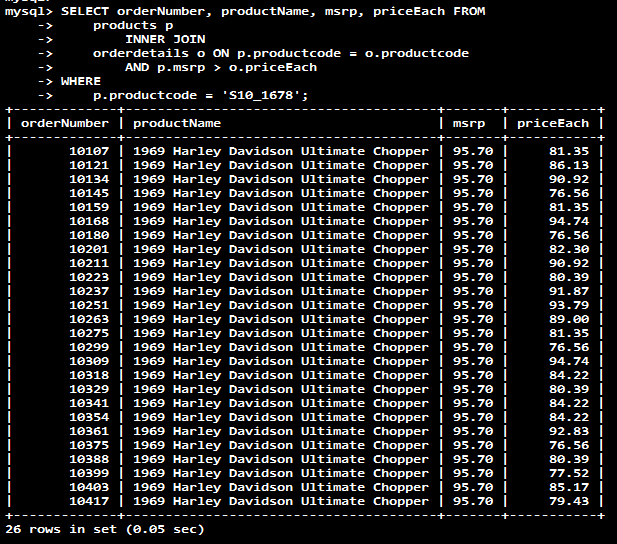
INNER JOIN

orderdetails o ON p.productcode = o.productcode

AND p.msrp > o.priceEach

WHERE

p.productcode = 'S10\_1678';



-- Qry : LEFT JOIN

-- Q : find all orders that belong to each customer ?

-- Remember :

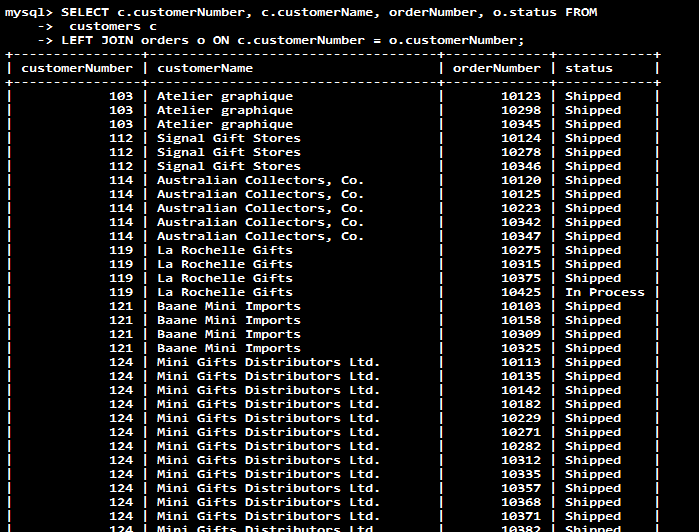
-- Each order in the orders table must belong to a customer in the customers table.

-- Each customer in the customers table can have zero or more orders in the orders table.

SELECT c.customerNumber, c.customerName, orderNumber, o.status FROM

customers c

LEFT JOIN orders o ON c.customerNumber = o.customerNumber;



-- Qry : LEFT JOIN + WHERE + IS NULL

-- Q : find all customers who have not placed any order ?

SELECT c.customerNumber, c.customerName, orderNumber, o.status

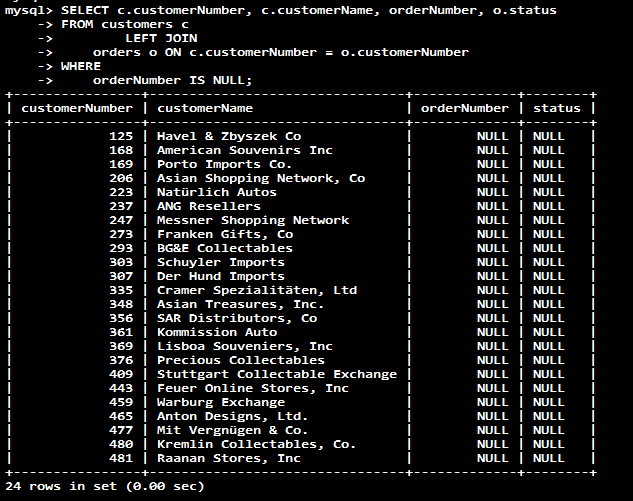
FROM customers c

LEFT JOIN

orders o ON c.customerNumber = o.customerNumber

WHERE

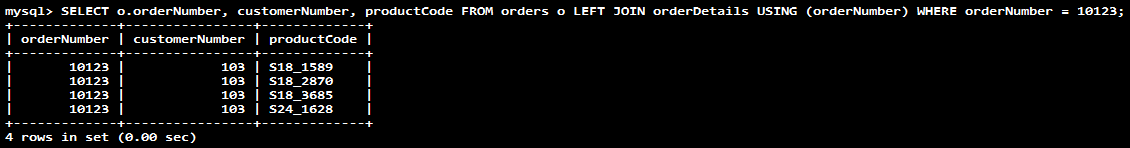
orderNumber IS NULL;



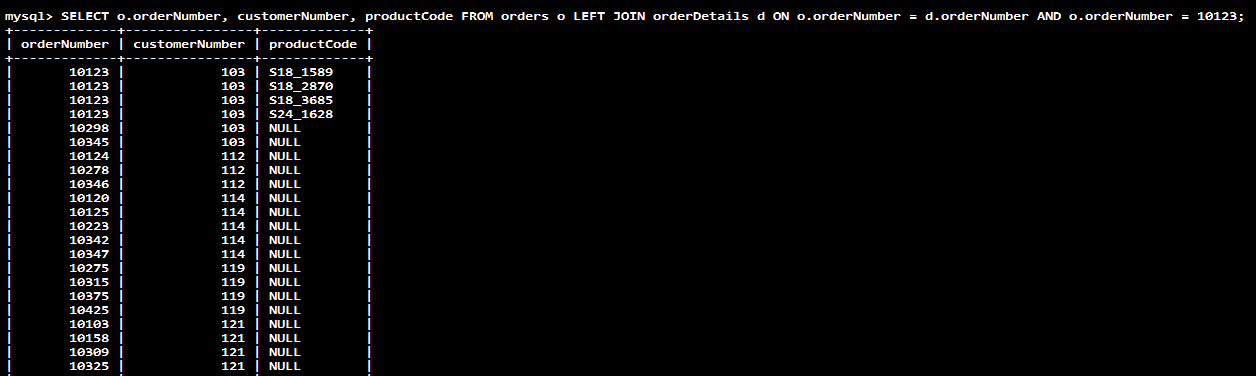
-- Qry : LEFT JOIN + WHERE VS ON

Q : Expliquer ces 2 Qry ?

SELECT o.orderNumber, customerNumber, productCode FROM orders o LEFT JOIN orderDetails USING (orderNumber) WHERE orderNumber = 10123;



SELECT o.orderNumber, customerNumber, productCode FROM orders o LEFT JOIN orderDetails d ON o.orderNumber = d.orderNumber AND o.orderNumber = 10123;



-- Qry : RIGHT JOIN

-- Q : get the sales representatives and their customers ?

SELECT concat(e.firstName,' ', e.lastName) salesman, e.jobTitle, customerName

FROM

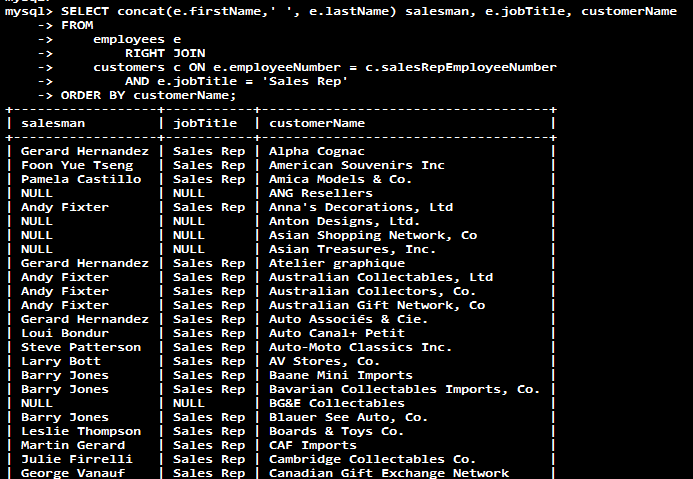
employees e

RIGHT JOIN

customers c ON e.employeeNumber = c.salesRepEmployeeNumber

AND e.jobTitle = 'Sales Rep'

ORDER BY customerName;



-- Qry : SELF JOIN

-- Q : Display a list of customers who locate in the same city ?

SELECT

c1.city, c1.customerName, c2.customerName

FROM

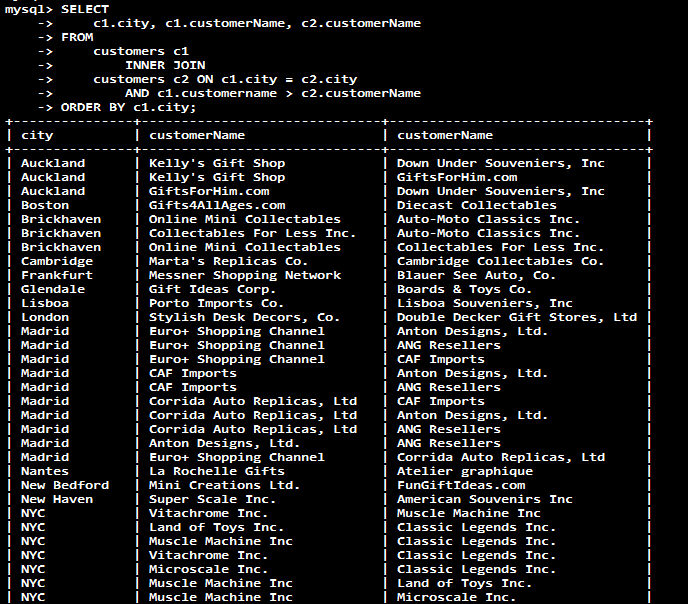
customers c1

INNER JOIN

customers c2 ON c1.city = c2.city

AND c1.customername > c2.customerName

ORDER BY c1.city;



-- Q : get the whole organization structure (organigramme de la société) ?

SELECT

CONCAT(m.lastname, ', ', m.firstname) AS 'Manager',

CONCAT(e.lastname, ', ', e.firstname) AS 'Direct report'

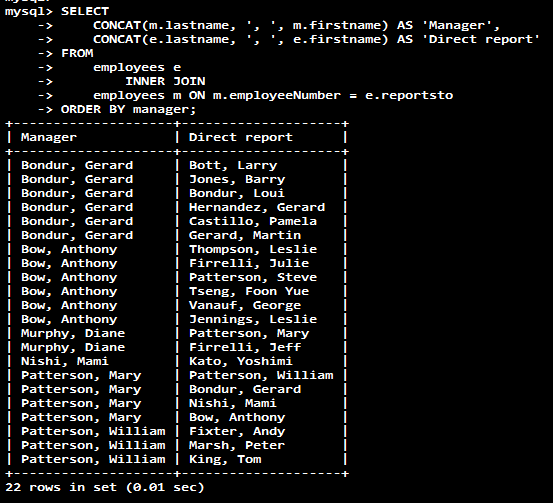
FROM

employees e

INNER JOIN

employees m ON m.employeeNumber = e.reportsto

ORDER BY manager;



-- Qry : SELF JOIN + IFNULL

-- La dernière Qry fonctionne bien sauf que ns avons juste les noms des fonctionnaires (elle ne retourne pas le nom des managers alorsq ns avons besoin de l'organigramme de la société)

-- R : get the whole organization structure ?

SELECT

IFNULL(CONCAT(m.lastname, ', ', m.firstname), -- retourner le nom du manager

'Top Manager') AS 'Manager',

CONCAT(e.lastname, ', ', e.firstname) AS 'Direct report'

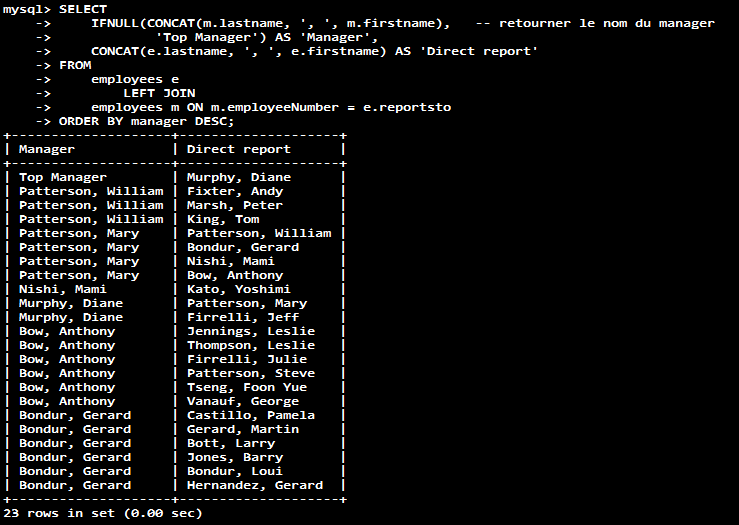
FROM

employees e

LEFT JOIN

employees m ON m.employeeNumber = e.reportsto

ORDER BY manager DESC;



-- Qry : CROSS JOIN

Un exemple :

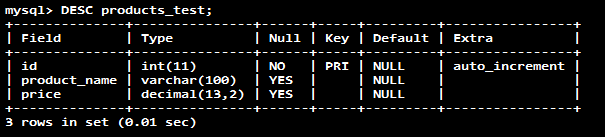
CREATE TABLE products**\_test** (

id INT PRIMARY KEY AUTO\_INCREMENT,

product\_name VARCHAR(100),

price DECIMAL(13 , 2 )

);

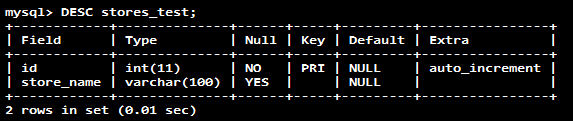


CREATE TABLE stores**\_test** (

id INT PRIMARY KEY AUTO\_INCREMENT,

store\_name VARCHAR(100)

);



CREATE TABLE sales**\_test** (

product\_id INT,

store\_id INT,

quantity DECIMAL(13 , 2 ) NOT NULL,

sales\_date DATE NOT NULL,

PRIMARY KEY (product\_id , store\_id),

FOREIGN KEY (product\_id)

REFERENCES products (id)

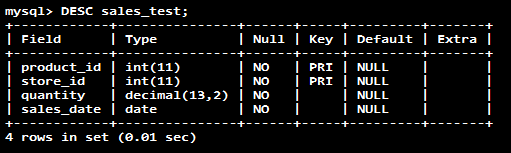
ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (store\_id)

REFERENCES stores (id)

ON DELETE CASCADE ON UPDATE CASCADE

);



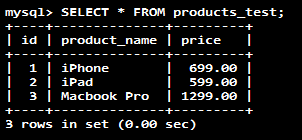
-- Suppose we have three products iPhone, iPad and Macbook Pro which are sold in two stores North and South.

INSERT INTO products(product\_name, price)

VALUES('iPhone', 699),

('iPad',599),

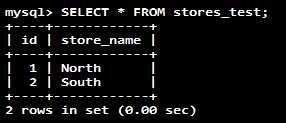
('Macbook Pro',1299);



INSERT INTO stores(store\_name)

VALUES('North'),

('South');



INSERT INTO sales(store\_id,product\_id,quantity,sales\_date)

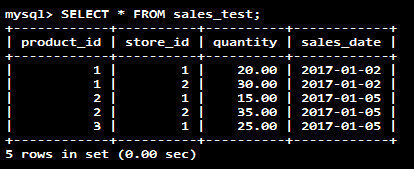
VALUES(1,1,20,'2017-01-02'),

(1,2,15,'2017-01-05'),

(1,3,25,'2017-01-05'),

(2,1,30,'2017-01-02'),

(2,2,35,'2017-01-05');



-- Q : get the total sales for each store and for each product, you calculate the sales and group them by store and product as follows ?

SELECT

store\_name,

product\_name,

SUM(quantity \* price) AS revenue

FROM

Sales**\_test**

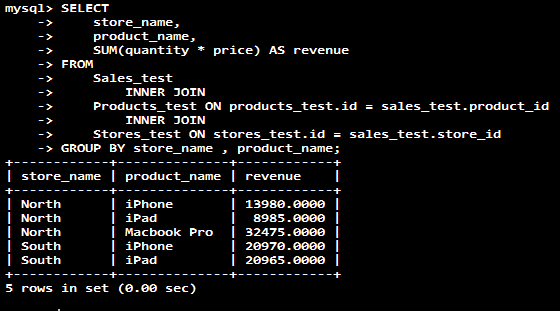
INNER JOIN

Products**\_test** ON products**\_test**.id = sales**\_test**.product\_id

INNER JOIN

Stores**\_test** ON stores**\_test**.id = sales**\_test**.store\_id

GROUP BY store\_name , product\_name;



-- Q : you want to know also which store had no sales of a specific product ?

-- First, you use the CROSS JOIN clause to get the combination of all stores and products:

SELECT

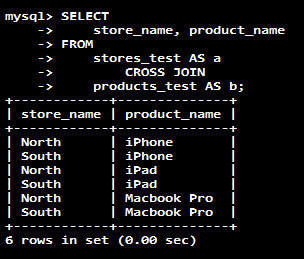
store\_name, product\_name

FROM

stores**\_test** AS a

CROSS JOIN

products**\_test** AS b;



-- Next, you join the result of the query above with the query that returns the total of sales by store and by product. The following query illustrates the idea

SELECT

b.store\_name,

a.product\_name,

IFNULL(c.revenue, 0) AS revenue

FROM

products**\_test** AS a

CROSS JOIN

stores**\_test** AS b

LEFT JOIN

(SELECT

stores**\_test**.id AS store\_id,

products**\_test**.id AS product\_id,

store\_name,

product\_name,

ROUND(SUM(quantity \* price), 0) AS revenue

FROM

sales**\_test**

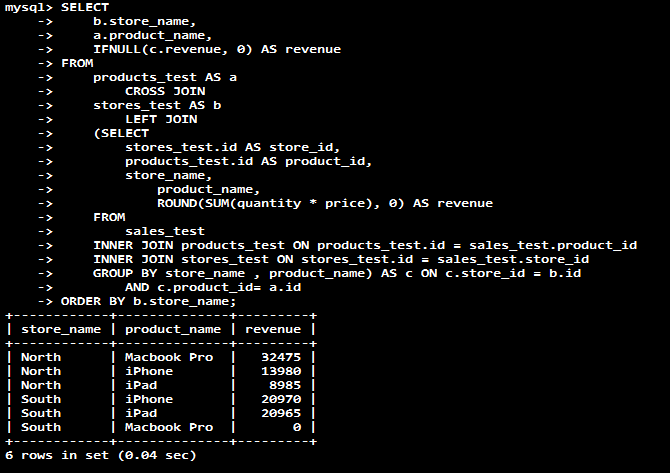
INNER JOIN products**\_test** ON products**\_test**.id = sales**\_test**.product\_id

INNER JOIN stores**\_test** ON stores**\_test**.id = sales**\_test**.store\_id

GROUP BY store\_name , product\_name) AS c ON c.store\_id = b.id

AND c.product\_id= a.id

ORDER BY b.store\_name;



**FIN**