FINAL PROJECT REPORT:

THEME : TRADING COMPANY DATABASE

NAME : **LE Thu Huong**
CLASS : **ADEO1 _ Relational Database**
PROFESSOR : **Lyes Touati**
SUBMITTION DATE : **13/12/2017**

TABLE OF CONTENTS

1 INTRODUCTION:

Current problem of ABC trading company:

Currently, they are using a traditional way of information's storage: employees keep tract of their order's by fax paper and input into excel files. Each sale-employee does it individually, then, they required to submit their list at the end of the day to gather all information. These cause many problem such as overlapping number of order, incorrect product's name, missing information,ect; which lead to inefficient business performances and low productivity results. Moreover, it brings dissatisfaction to our clients because of wrong product or order, slow transaction responding.

Key solution to slove the problem:

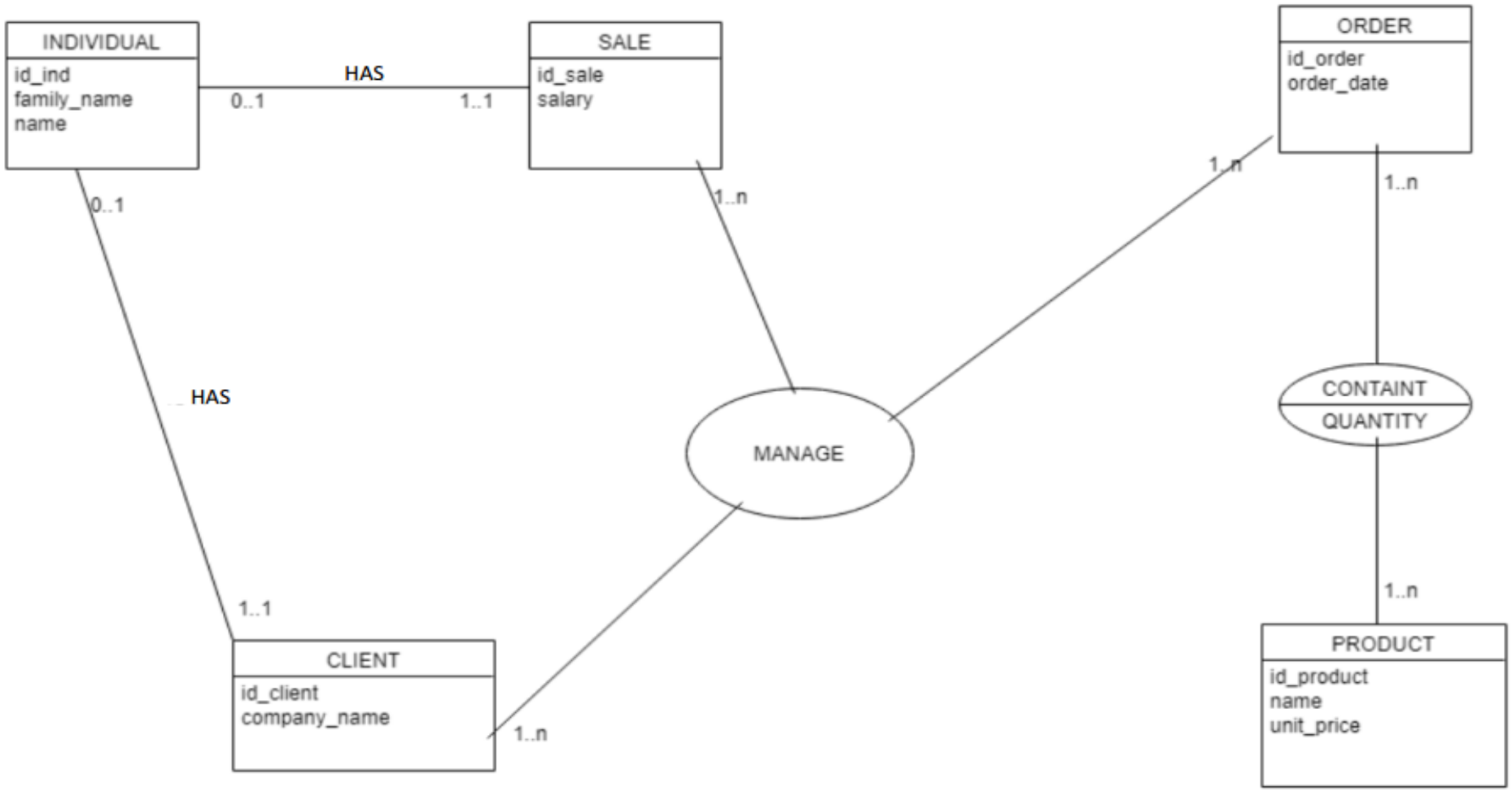
One of the company's priority solution is to build up their database system to better keep tract of transaction with clients, speed action to order, if needed.

This paper will help ABC company to build up their database: (1) model the actual objects by conceptual data model (2) transform into logical data model, apply normalization rules (3) build up table creation script in database (4) insertion and updating all information (5) manipulating database to better business performances

2 CONCEPTUAL DATA MODEL:

Based on the above ideals, we re - defined our problem by building conceptual data model, which is a useful tool to model coherence and limit redundancy

Diagram 1: Conceptual Data Model



3 LOGICAL DATA MODEL:

Next step is a tranformation from conceptual data model to logical data model based on its conversion rules. This is an important approach to SQL transcription.

Individual (id_ind , family_name, name)
Sale (id_sale , # id_ind , salary)
Client (id_client , # id_ind , company_name)
Order (id_order , order_date)
Manage(#id_order,#id_sale,#id_client)
Products (id_product , name, unit_price)
Contain (#id_order, #id_product, quantity)

4 SQL – Data Definition Language: CREATE, DROP, ALTER table

The database consists of 7 tables. We define each and every tables as below:

SQL Script	Explanation/ Result
DROP TABLE	

`DROP TABLE if EXISTS individual CASCADE;`
`DROP TABLE if EXISTS sale CASCADE;`
`DROP TABLE if EXISTS client CASCADE;`
`DROP TABLE if EXISTS oorder CASCADE;`
`DROP TABLE if EXISTS manage CASCADE;`
`DROP TABLE if EXISTS product CASCADE;`
`DROP TABLE if EXISTS contain CASCADE;`
	DROP TABLE : Remove table individual, sale, client, oorder, manage, product, contain if they exist in the database.
	IF EXISTS: Do not throw an error if the table does not exist.
	CASCADE: Automaticcally drop objects that depend on the table

SQL Script	Explanation/ Result
TABLE individual consists all individual 's information	

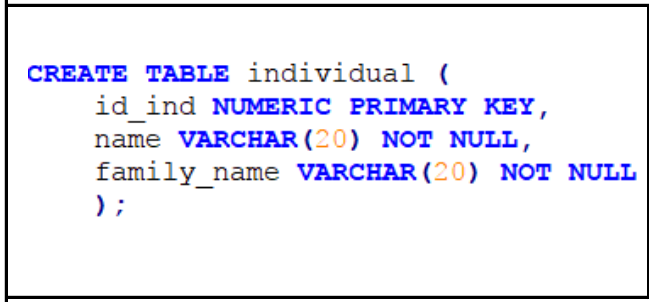



TABLE sale consists all sale person 's information	

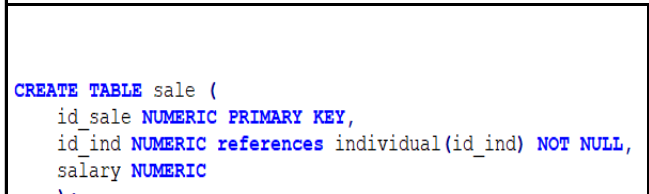
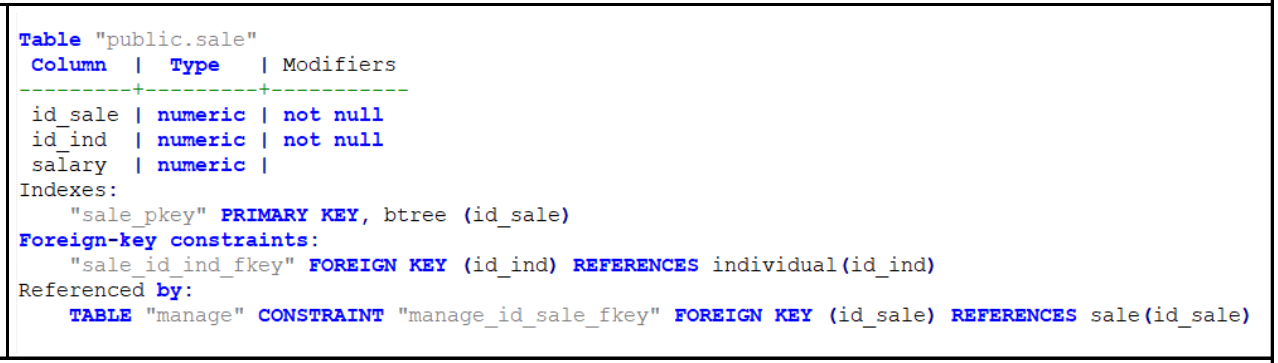
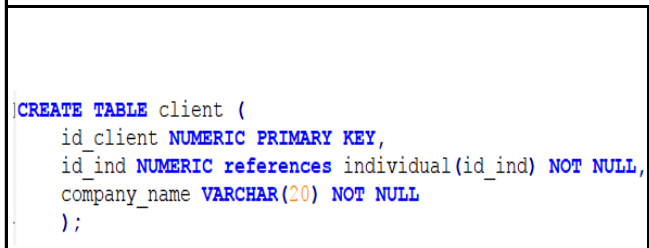
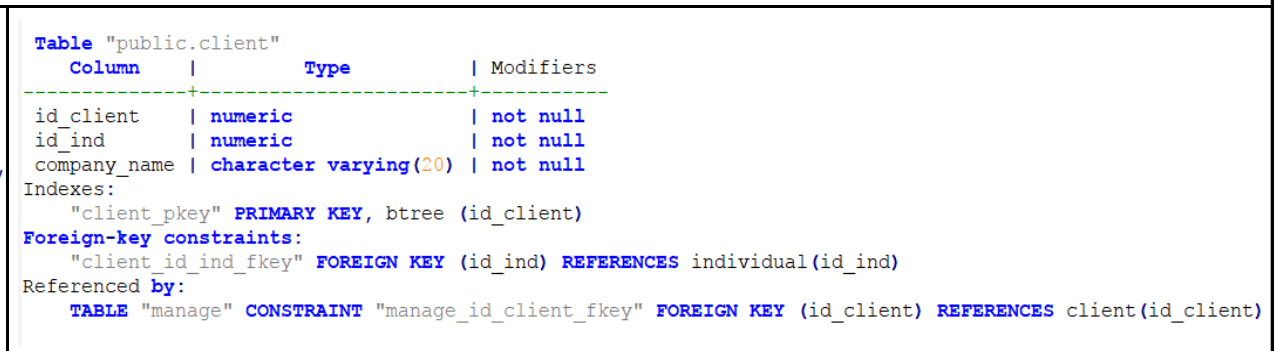

	

TABLE client consists all client's information	
	

SQL Script	Explanation/ Result
TABLE oorder consists all order's date and number	

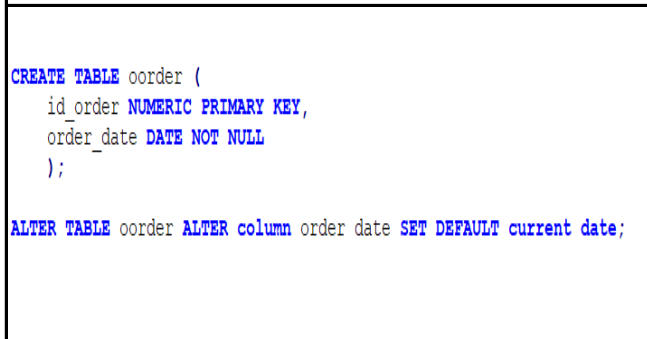
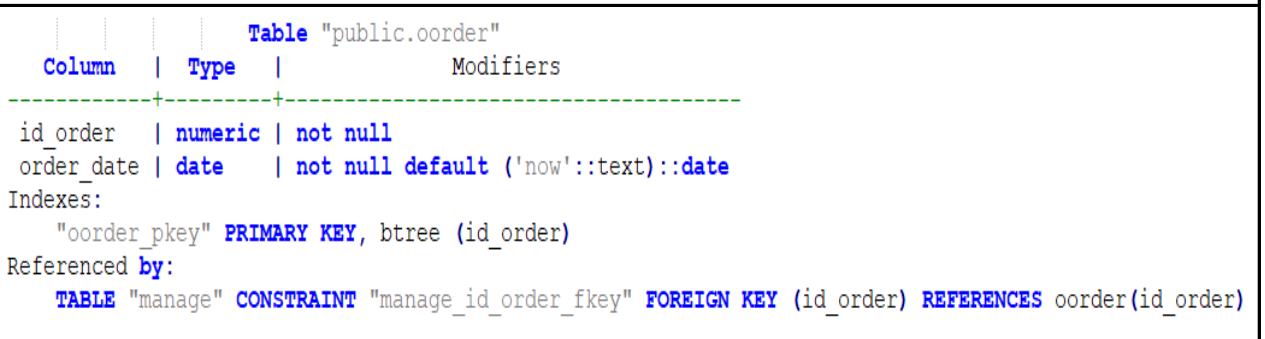
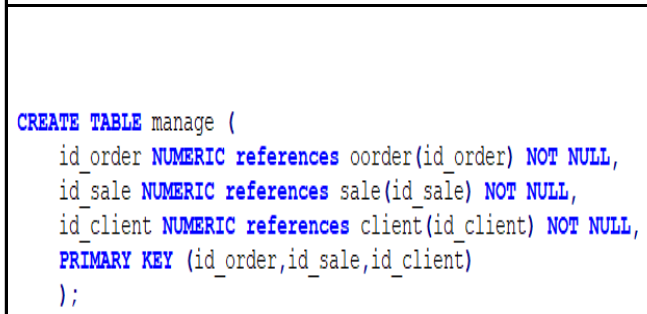
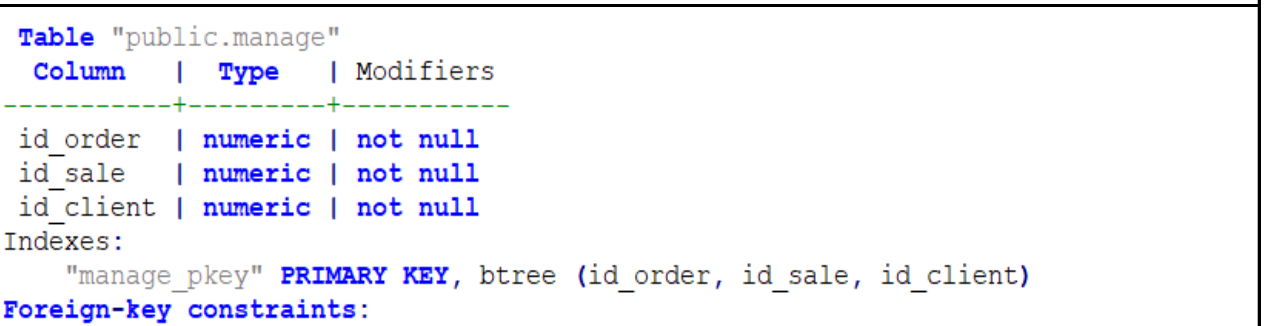

	

TABLE manage consists each and every order's number, sale person in - charged and its client	
	

SQL Script	Explanation/ Result
TABLE product consists product's information, its identify number, name and its price	

`CREATE TABLE product (`
`id_product NUMERIC PRIMARY KEY,`
`pname VARCHAR(50) NOT NULL,`
`unit_price NUMERIC NOT NULL`
`);`
	`Table "public.product"`
	`Column | Type | Modifiers`
	`id_product | numeric | not null`
	`pname | character varying(50) | not null`
	`unit_price | numeric | not null`
	`Indexes:`
	`"product_pkey" PRIMARY KEY, btree (id_product)`
	
	`CREATE TABLE contain (`
	`id_order NUMERIC references oorder(id_order) NOT NULL,`
	`id_product NUMERIC references product(id_product) NOT NULL,`
	`PRIMARY KEY (id_order,id_product),`
	`quantity INTEGER`
	`);`

TABLE contain consists all order number, product number and its quantities	

`CREATE TABLE contain (`
`id_order NUMERIC references oorder(id_order) NOT NULL,`
`id_product NUMERIC references product(id_product) NOT NULL,`
`PRIMARY KEY (id_order,id_product),`
`quantity INTEGER`
`);`
	`Table "public.contain"`
	`Column | Type | Modifiers`
	`id_order | numeric | not null`
	`id_product | numeric | not null`
	`quantity | integer |`
	`Indexes:`
	`"contain_pkey" PRIMARY KEY, btree (id_order, id_product)`

5 SQL – Data Manipulation Language: INSERT...etc.

Build up our database by insertion all information from existing sources (paper, excel data) to the database for better storage, limit redundancy,…ect

SQL Script	Explanation/ Result
TABLE individual	
--TABLE INDIVIDUAL--(14) INSERT INTO individual VALUES (1,'Andrea','Santamaria'); INSERT INTO individual VALUES (2,'Ary','Abittan'); INSERT INTO individual VALUES (3,'Francois','Demaison'); INSERT INTO individual VALUES (4,'Leon','Vauban'); INSERT INTO individual VALUES (5,'Mathieu','Vasseur'); INSERT INTO individual VALUES (6,'Pierre','Niney'); INSERT INTO individual VALUES (7,'Alice','Fursac'); INSERT INTO individual VALUES (8,'Ana','Girardot'); INSERT INTO individual VALUES (9,'Charles','Chaplin'); INSERT INTO individual VALUES (10,'Adenoid','Hynkel'); INSERT INTO individual VALUES (11,'Emma','Stone'); INSERT INTO individual VALUES (12,'Mia','Lalaland'); INSERT INTO individual VALUES (13,'Ryan','Gosling'); INSERT INTO individual VALUES (14,'Sebastian','Yazz');	id_ind | name | family_name 1 | Andrea | Santamaria 2 | Ary | Abittan 3 | Francois | Demaison 4 | Leon | Vauban 5 | Mathieu | Vasseur 6 | Pierre | Niney 7 | Alice | Fursac 8 | Ana | Girardot 9 | Charles | Chaplin 10 | Adenoid | Hynkel 11 | Emma | Stone 12 | Mia | Lalaland 13 | Ryan | Gosling 14 | Sebastian | Yazz (14 rows)
TABLE sale	
--TABLE SALE--(7) INSERT INTO sale VALUES (1,1,3500); INSERT INTO sale VALUES (2,3,3000); INSERT INTO sale VALUES (3,6,4000); INSERT INTO sale VALUES (4,8,3600); INSERT INTO sale VALUES (5,9,5500); INSERT INTO sale VALUES (6,11,4500); INSERT INTO sale VALUES (7,13,4300);	id_sale | id_ind | salary 1 | 1 | 3500 2 | 3 | 3000 3 | 6 | 4000 4 | 8 | 3600 5 | 9 | 5500 6 | 11 | 4500 7 | 13 | 4300 (7 rows)

SQL Script	Explanation/ Result
TABLE client	
--TABLE CLIENT-(7) INSERT INTO client VALUES (2017,2,'Coco'); INSERT INTO client VALUES (2018,4,'Paris'); INSERT INTO client VALUES (2019,5,'Paris'); INSERT INTO client VALUES (2020,7,'Paris'); INSERT INTO client VALUES (2021,10,'Monaco'); INSERT INTO client VALUES (2022,12,'Lalaland'); INSERT INTO client VALUES (2023,14,'Lalaland');	id_client | id_ind | company_name ----------+--------+-------------- 2017 | 2 | Coco 2018 | 4 | Paris 2019 | 5 | Paris 2020 | 7 | Paris 2021 | 10 | Monaco 2022 | 12 | Lalaland 2023 | 14 | Lalaland (7 rows)
TABLE oorder	
--TABLE OORDER--(7) INSERT INTO oorder VALUES (241217,TO_DATE('24/12/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (441017,TO_DATE('04/10/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (770917,TO_DATE('07/09/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (441117,TO_DATE('04/11/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (210817,TO_DATE('21/08/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (310717,TO_DATE('31/07/2017','dd/mm/yyyy')); INSERT INTO oorder VALUES (312717,TO_DATE('31/07/2017','dd/mm/yyyy'));	id_order | order_date ----------+------------ 241217 | 2017-12-24 441017 | 2017-10-04 770917 | 2017-09-07 441117 | 2017-11-04 210817 | 2017-08-21 310717 | 2017-07-31 312717 | 2017-07-31 (7 rows)
TABLE manage	
--TABLE MANAGE--(7) INSERT INTO manage VALUES (241217,1,2017); INSERT INTO manage VALUES (441017,3,2018); INSERT INTO manage VALUES (770917,3,2019); INSERT INTO manage VALUES (441117,4,2020); INSERT INTO manage VALUES (210817,5,2021); INSERT INTO manage VALUES (310717,6,2022); INSERT INTO manage VALUES (312717,7,2023);	id_order | id_sale | id_client ----------+---------+----------- 241217 | 1 | 2017 441017 | 3 | 2018 770917 | 3 | 2019 441117 | 4 | 2020 210817 | 5 | 2021 310717 | 6 | 2022 312717 | 7 | 2023 (7 rows)

SQL Script	Explanation/ Result
TABLE product	
--TABLE PRODUCT--(7) INSERT INTO product VALUES (100,'Guitar','1000.99'); INSERT INTO product VALUES (150,'Journal intime','20.88'); INSERT INTO product VALUES (151,'Sable noir livre','55.88'); INSERT INTO product VALUES (152,'Vetement femme','500.88'); INSERT INTO product VALUES (200,'Noir et blanc film','25.99'); INSERT INTO product VALUES (160,'Voyage aux etats unis','5000.88');	id_product | pname | unit_price 100 | Guitar | 1000.99 150 | Journal intime | 20.88 151 | Sable noir livre | 55.88 152 | Vetement femme | 500.88 200 | Noir et blanc film | 25.99 160 | Voyage aux etats unis | 5000.88 (6 rows)
TABLE contain	
--TABLE CONTAIN--(7) INSERT INTO contain VALUES (241217,100,1); INSERT INTO contain VALUES (441017,150,2); INSERT INTO contain VALUES (770917,151,1000); INSERT INTO contain VALUES (441117,100,2); INSERT INTO contain VALUES (210817,200,20); INSERT INTO contain VALUES (310717,160,1); INSERT INTO contain VALUES (312717,160,1);	id_order | id_product | quantity 241217 | 100 | 1 441017 | 150 | 2 770917 | 151 | 1000 441117 | 100 | 2 210817 | 200 | 20 310717 | 160 | 1 312717 | 160 | 1 (7 rows)

5 SQL – Data Manipulation Language: SELECT, VIEW...etc.

Company uses database to improve business performance, keep track of basic transaction,…ect; but it is also provide information that will help the decision making more efficiently, help managers and employees make better decisions.

SELECT QUERY	Explanation/ Result
Sort all product price in ascending order. We want to see its name and its price as the result	
`SELECT pname,unit_price FROM product` `ORDER BY unit_price ASC;`	pname | unit_price Journal intime | 20.88 Noir et blanc film | 25.99 Sable noir livre | 55.88 Vetement femme | 500.88 Guitar | 1000.99 Voyage aux etats unis | 5000.88 (6 rows)
How many orders ABC company received in year 2017?	
`SELECT COUNT(id_order) AS total_order_2017 FROM oorder` `WHERE DATE_PART('year',order_date)=2017;`	total_order_2017 7 (1 row)
What is the most expensive product?	

`SELECT * FROM product`
`WHERE unit_price = (`
`SELECT MAX (unit_price) FROM product);`
	id_product | pname | unit_price
	160 | Voyage aux etats unis | 5000.88
	(1 row)

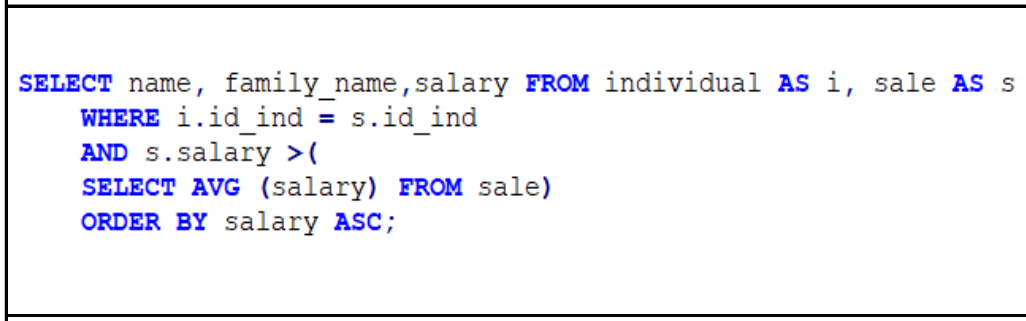
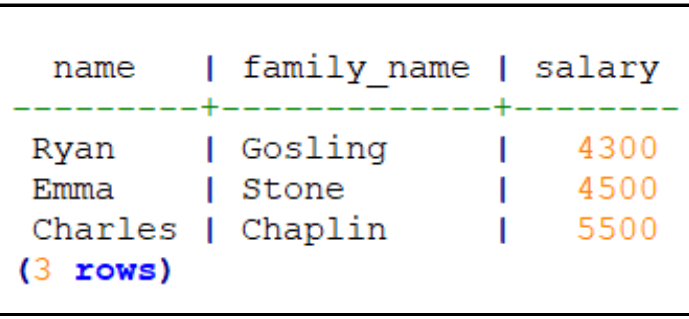
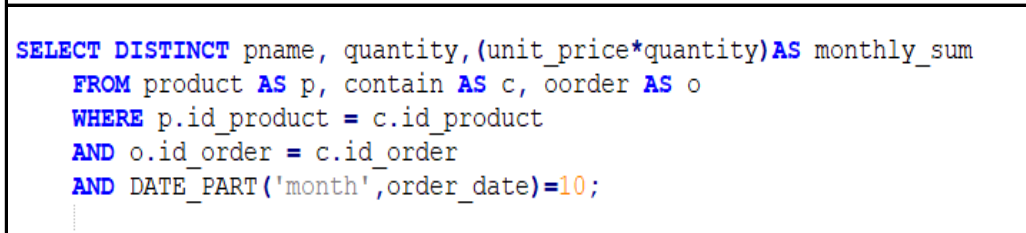
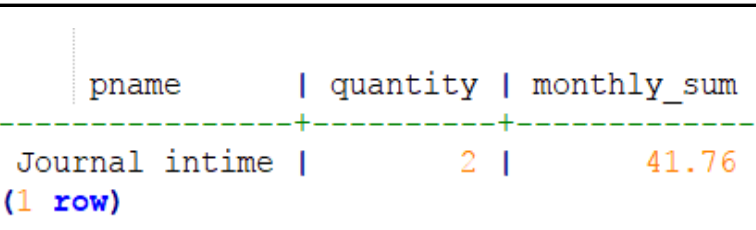
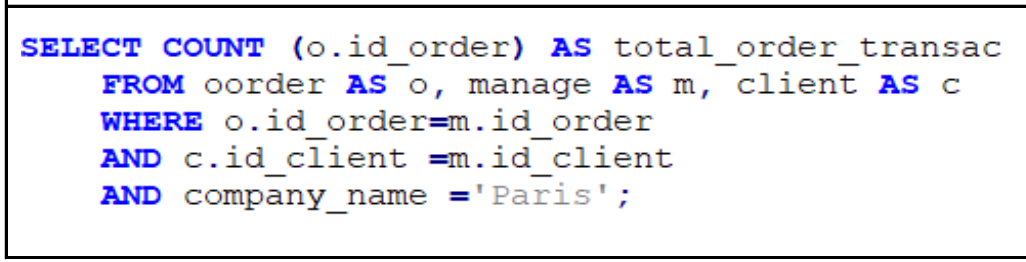
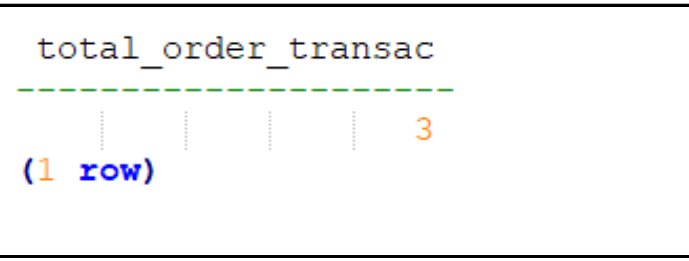
SELECT QUERY	Explanation/ Result
Sort all employees' name in ascending order, we want their name, family_name, id and salary in the result	

`SELECT name, family_name, id_sale, salary FROM individual`
`NATURAL JOIN sale`
`ORDER BY name ASC;`
	name | family_name | id_sale | salary
	Ana | Girardot | 4 | 3600
	Andrea | Santamaria | 1 | 3500
	Charles | Chaplin | 5 | 5500
	Emma | Stone | 6 | 4500
	Francois | Demaison | 2 | 3000
	Pierre | Niney | 3 | 4000
	Ryan | Gosling | 7 | 4300
	(7 rows)

who are our clients in Lalaland company? We want to know their name and family_name to contact.	

`SELECT name, family_name FROM individual AS i`
`INNER JOIN client AS cl ON i.id_ind=cl.id_ind`
`AND company_name ='Lalaland';`
	name | family_name
	Mia | Lalaland
	Sebastian | Yazz
	(2 rows)

What is the total sum of all orders until today?	
`SELECT SUM( unit_price * quantity) AS total_sum` `FROM product NATURAL JOIN contain;`	total_sum 69446.29 (1 row)

SELECT QUERY	Explanation/ Result
Who is/are the employees, that have salary higher than the average's one?	
	
What products sold in Octorber? Display the Octorber's sale result?	
	
How many orders have we received from Paris company?	
	

SELECT QUERY	Explanation/ Result
What is the earliest date of order from Paris company to us?	
SELECT MIN (order_date) **AS** first_transac_date_w_paris **FROM** oorder **AS** o,manage **AS** m, client **AS** c **WHERE** o.id_order=m.id_order **AND** c.id_client =m.id_client **AND** company_name ='Paris';	first_transac_date_w_paris ---------- 2017-09-07 (1 row)
Display total order of each individual to our companry.	
SELECT i.name, i.family_name, **COUNT**(m.id_order) **AS** number_of_order **FROM** individual **AS** i **LEFT JOIN** client **AS** cl **ON** i.id_ind = cl.id_ind **LEFT JOIN** manage **AS** m **ON** cl.id_client = m.id_client **GROUP BY** i.name, i.family_name **ORDER BY** number_of_order **DESC**;	name | family_name | number_of_order ----------+-------------+----------------- Leon | Vauban | 1 Mia | Lalaland | 1 Mathieu | Vasseur | 1 Adenoid | Hynkel | 1 Alice | Fursac | 1 Ary | Abittan | 1 Sebastian | Yazz | 1 Charles | Chaplin | 0 Pierre | Niney | 0 Andrea | Santamaria | 0 Francois | Demaison | 0 Ana | Girardot | 0 Emma | Stone | 0 Ryan | Gosling | 0 (14 rows)

<table>
<tr><th>SELECT QUERY</th><th>Explanation/ Result</th></tr>
<tr><td colspan="2">Who is the client have purchased all products from our company?</td></tr>
<tr><td>

```
SELECT i.id_ind, name, family_name
    FROM individual AS i, client AS cl,manage AS m,
    contain AS c, product AS p
    WHERE i.id_ind = cl.id_ind
    AND cl.id_client = m.id_client
    AND m.id_order = c.id_order
        GROUP BY i.id_ind
    HAVING COUNT (DISTINCT c.id_product)=(
    SELECT COUNT (p.id_product) FROM product AS p);
```

</td><td>There is no client, who have purchased all company products

```
 id_ind | name | family_name
--------+------+-------------
(0 rows)
```

</td></tr>
<tr><td colspan="2">Create view to see all clients in Paris company</td></tr>
<tr><td>

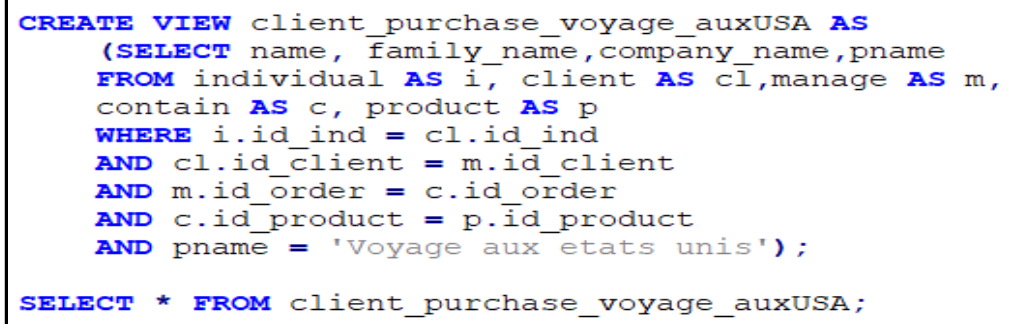
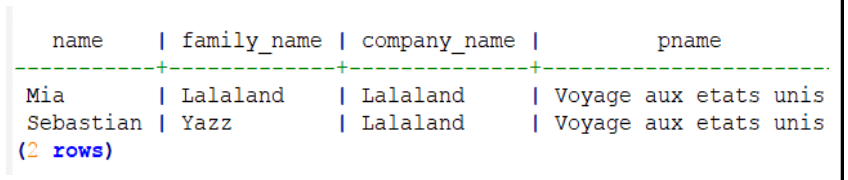
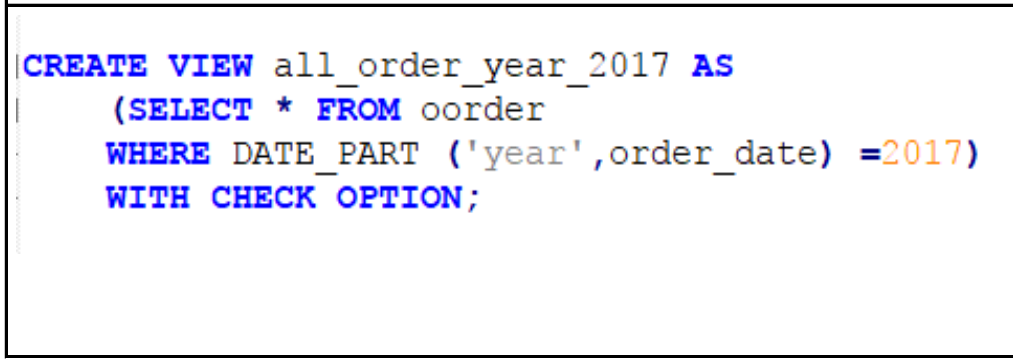
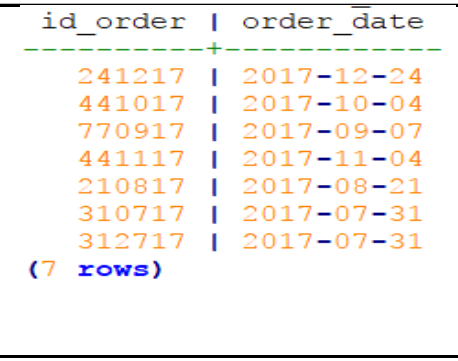
```
CREATE VIEW paris_company_client AS
    (SELECT name, family_name,company_name
    FROM individual AS i, client AS cl
    WHERE i.id_ind = cl.id_ind
    AND company_name = 'Paris');

SELECT * FROM paris_company_client;
```

</td><td>

```
  name    | family_name | company_name
----------+-------------+--------------
 Leon     | Vauban      | Paris
 Mathieu  | Vasseur     | Paris
 Alice    | Fursac      | Paris
(3 rows)
```

</td></tr>
</table>

SELECT QUERY	Explanation/ Result
Create view of all client have purchased 'Voyage aux etats unis'	
	
Create view of all order in year of 2017. Use check option to avoid all updates, which not in year 2017.	
	

6 CONCLUSION

We created a database that ABC company can use for keeping track on its business, improve the business performance. ABC company data management system has been more effective, long durability, higher consistency. Of course, the works also get easier, when the company can use a database on a computer, rather than managing everything on traditional way of doing.

Moreover, the database structure is quite simple, which makes it easy for other employees to work and understand it.

This project gave us the opportunity to try our new skills in practice: Thanks to intuitive queries and language, we can even manipulate a great deal of database in system. We gained deeper understanding on database model and design. It will be helpful for decision making in real life situations.