Mandatory2 Relational database and SQL

Pablo Martín García, Aaron Reboredo Vázquez

Mandatory assignment 2:

Relational database and SQL

Part 1:

Thinking on the purpose of this assignment and its nature we are going to use Excel for this first part in order to illustrate our explanation. In this way we can also show how the tables evolve and show the answers and the tasks in a more visual way.

The next table is the same as we have on the assignment PDF.

Cust-name	Cust addr	City	Phone	Order ID	Order Date	Prod_id	Descr	Unit_Price	Qty	Empl	Phone
Donal Duck	Strandveien 38	7713 Steinkjer	743210	110	07/01/2018	12 + 13	Pencil+Rubber	5+2	50+10	Al Pacino	987654
Scrooge McDuck	Kongens 39	7713 Steinkjer	743211	111	07/01/2018	12+14	Pencil+Ruler	5+6	1+1	Julia Roberts	987655
Daisy Duck	Strandveien 38	7713 Steinkjer	743212	112	07/02/2018	15	Sharp	7	1	Tom Cruise	987656
Thelma Duck	Hogskolevegen 27	7600 Levanger	744321	113	07/02/2018	12+13	Pencil+Rubber	5+2	10+10	Al Pacino	987654

Thinking about data bases is think about normalization and optimization according some rules. Some of them are the ones called Normal Forms. After modifying the table according the **first normal form** we get the nest table:

Cust-name	Cust addr	City	Phone	Order ID	Order Date	Prod_id	Descr	Unit_Price	Qty	Empl	Phone
Donal Duck	Strandveien 38	7713 Steinkjer	743210	110	07/01/2018	12	Pencil	5	50	Al Pacino	987654
Donal Duck	Strandveien 38	7713 Steinkjer	743210	110	07/01/2018	13	Rubber	2	10	Al Pacino	987654
Scrooge McDuck	Kongens 39	7713 Steinkjer	743211	111	07/01/2018	12	Pencil	5	1	Julia Roberts	987655
Scrooge McDuck	Kongens 39	7713 Steinkjer	743211	111	07/01/2018	14	Ruler	6	1	Julia Roberts	987655
Daisy Duck	Strandveien 38	7713 Steinkjer	743212	112	07/02/2018	15	Sharp	7	1	Tom Cruise	987656
Thelma Duck	Hogskolevegen 27	7600 Levanger	744321	113	07/02/2018	12	Pencil	5	10	Al Pacino	987654
Thelma Duck	Hogskolevegen 27	7600 Levanger	744321	113	07/02/2018	13	Rubber	2	10	Al Pacino	987654

Now we have to choose a primary key for our table, it has to be an unique identifier with no duplicates. We have to be sure that in is never going to be null and that always is going to have a value. We want to choose one to make the comparisons and analysis as simple as we can and if we think in terms of space choose an integer value is the best option. According to all of this the **Order_id** seems to be a good key vale. But thinking about the data stored on our table we realize that we need another key in order to have access to all the data, and identify any row on our table, **Product_id** is going to be the chosen one to make the combination of Primary Keys (**Order_id, Product_id**).

In combination with the **Product_id** we come up with a **pair of primary keys** that allow us to find information about every row on our tables.We can access to the information about the customers using the **Order_id** and information about the products using the **Product_id**. We will have access to the employee´s information using link tables like the Order_Employee table that contains the Product_id and Employees_id.

Having this in mind we can think in a link table using those primary keys:

PRODUCT-ORDER TABLE		
Product_id	Order_id	Qty
12	110	50
13	110	10
12	111	1
14	111	1
15	112	1
12	113	10
13	113	10

Adding here the quantities (Qty) we can have access to the information that we were not able to using those pair of primary keys, in this case the quantities of each product bought by the customers.

At this point we have to think on the functional **dependencies** and the **second and third normal forms** in order to divide the tables correctly:

The dependencies can be seen in the next graphic:

Functional dependencies:

(**Order_id, Product_id**) ->Qty

Partial dependencies:

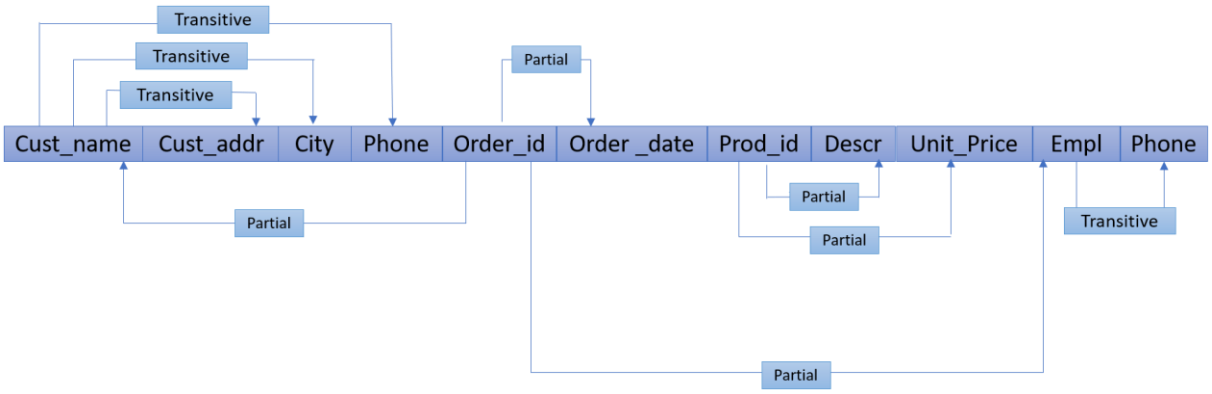
Order_id -> Order_date, Customer_name, Employee.

Product_id ->Descr, unit price.

Transitive dependencies:

Cust_name -> Address, City, Phone number.

Employee-> phone



Then, the division and the resulting tables according second and third form are going to be the next ones:

CUSTOMER TABLE						
Customer_id	Cust-name	Phone	Adress_Id	City_Id	Order_id	Order Date
1	Donal Duck	743210	1	1	110	07/01/2018
2	Scrooge McDuck	743211	2	1	111	07/01/2018
3	Daisy Duck	743212	1	1	112	07/02/2018
4	Thelma Duck	744321	3	2	113	07/02/2018

In our case we are going to make it even more simple and better in terms of size or weight for saving data and we are going to substitute some attributes for integers as identifiers. It is the case of Customer_id, Adress_Id and City_id.

CITY TABLE	
City_Id	City
1	7713 Steinkjer
2	7600 Levanger

ADRESS TABLE	
Adress_Id	Cust addr
1	Strandveien 38
2	Kongens 39
3	Hogskolevegen 27

EMPLOYEE TABLE		
Employee_Id	Empl	Phone
1	Al Pacino	987654
2	Julia Roberts	987655
3	Tom Cruise	987656

PRODUCT TABLE		
Product_id	Descr	Unit_Price
12	Pencil	5
13	Rubber	2
14	Ruler	6
15	Sharp	7

Link Table	ORDER EMPLOYEE TABLE
Order_id	Employee_Id
110	1
111	2
112	3
113	1

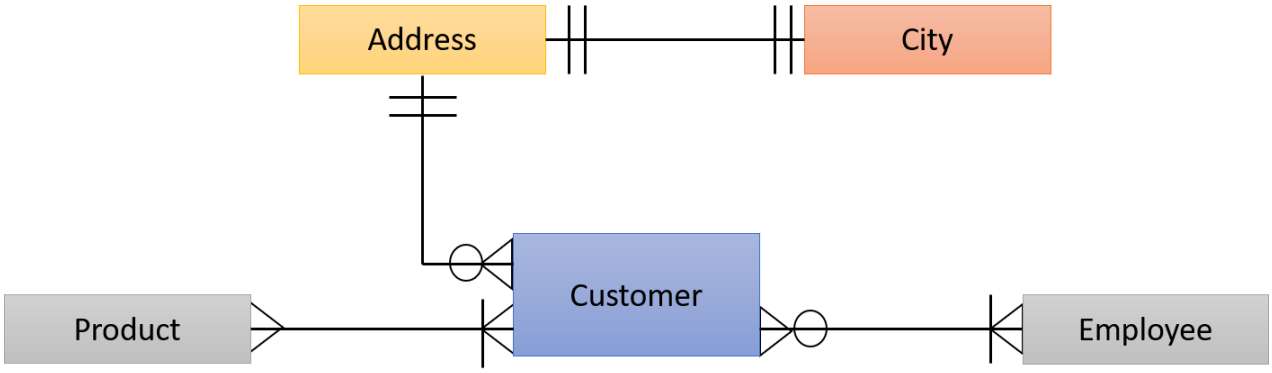
PRODUCT-ORDER TABLE		
Product_id	Order_id	Qty
12	110	50
13	110	10
12	111	1
14	111	1
15	112	1
12	113	10
13	113	10

Here we can see the tables and we realized that we can find the necessary data following the tables. Foreing keys are the ones are going to b very usefull in terms of building link tables.

The ones that are **green colored** are going to be the **primary keys** of their respective tables and the **yelow colored** ones are the **foreing keys**. Those last ones allow us to create link tables and make relations between them.

We can not forgive thath the primary keys of our original table and for the complete database are the combination about we were talking about:

Once we have decided and make the divisions and find the dependencies between values we can elaborate an ER-diagram showing the relationships between tables:



In this way and for our table distribution the **foreing keys** that have **interest** are :

- Customer_id that identifies each customer and relates Products table and quantities with the customer table.
- Product_id, that relates Products table and quantities with the Customer table and identifies each product .

- Adress_id, that relates Customer and Address tables.
- City_id, that relates Customer and City tables.
- Employee_id that relates Customer table with the employee asigned to each one (employees table).

Part two:

In order to work with data it is properly to use tools as SQL. We create queries in order to work an manage our data bases. The queries we have created in order to complete the tasks are the next ones:

- First of all we must insert our data and tables in Hera :

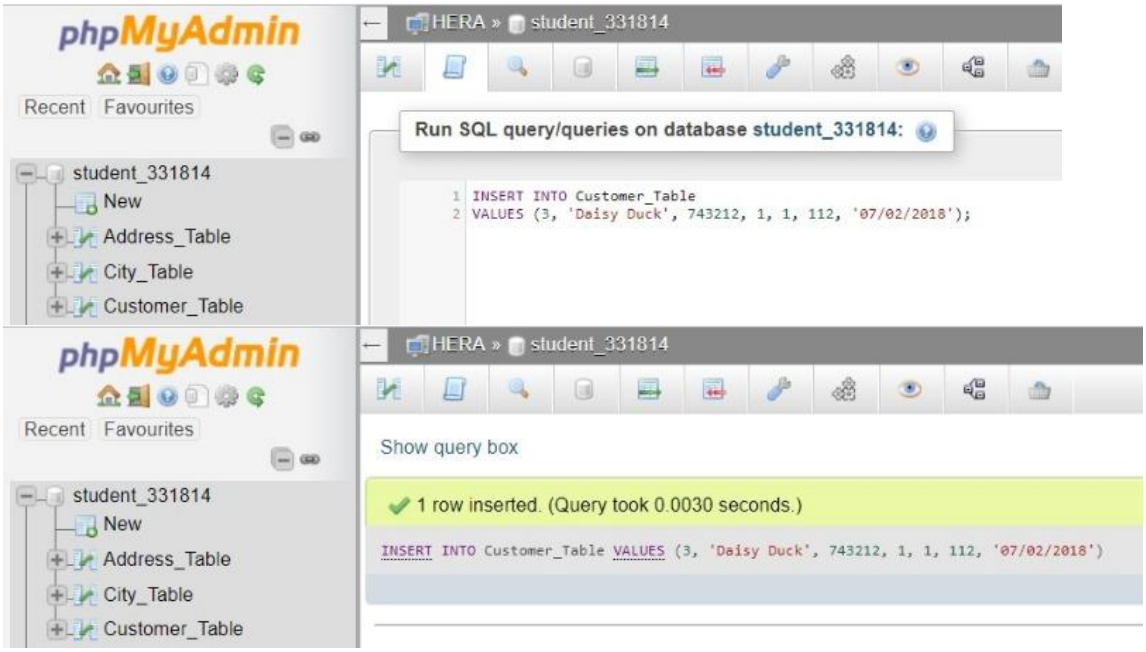
***We are going to upload the final tables on the user 331810 in Hera.**

MySQL queries:

CUSTOMER TABLE

```
/* Create table */
CREATE TABLE Customer_Table(
    Customer_id INT,
    Cust_name VARCHAR(255),
    Phone INT,
    Adrress_id INT,
    City_id INT,
    order_id INT,
    Order_date VARCHAR(255),
    PRIMARY KEY (order_id),
    FOREIGN KEY (Adrress_id) REFERENCES
Address_Table(Address_id),
    FOREIGN KEY (City_id) REFERENCES City_Table(City_id)
);
```

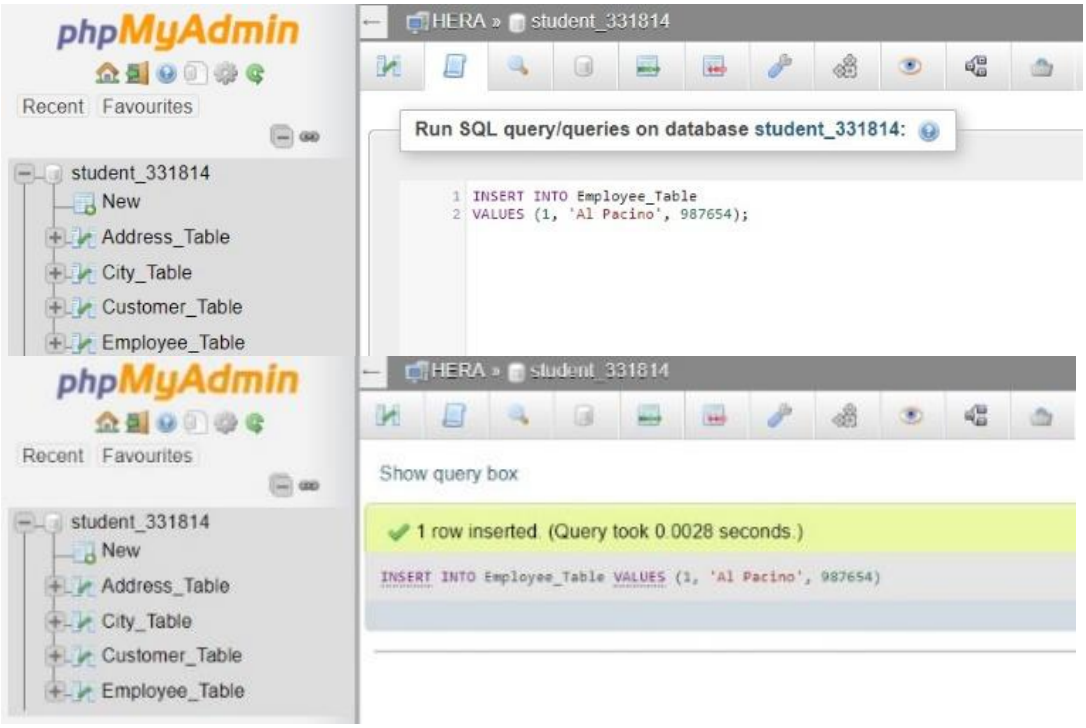
```
/* Insert values */
INSERT INTO Customer_Table
VALUES (1, 'Donald Duck', 743210, 1, 1, 110, '07/01/2018');

INSERT INTO Customer_Table
VALUES (2, 'Scrooge McDuck', 743211, 2, 1, 111, '07/01/2018');

INSERT INTO Customer_Table
VALUES (3, 'Daisy Duck', 743212, 1, 1, 112, '07/02/2018');

INSERT INTO Customer_Table
VALUES (4, 'Thelma Duck', 744321, 3, 2, 113, '07/02/2018');
```



Example of insertion on this table response.

EMPLOYEE TABLE

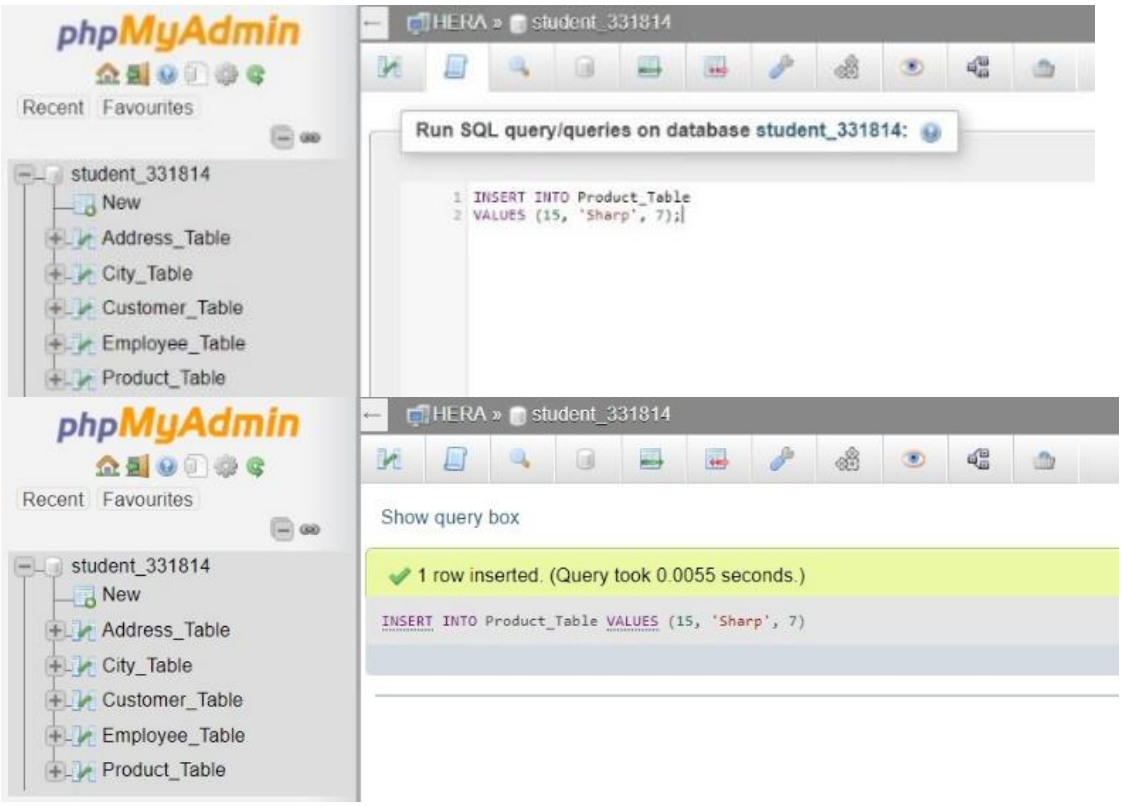
```
/* Create table */
CREATE TABLE Employee_Table(
    Employee_id INT,
    Empl VARCHAR(255),
    Phone INT,
    PRIMARY KEY (Employee_id)
);

/* Insert values */
INSERT INTO Employee_Table
VALUES (1, 'Al Pacino', 987654);

INSERT INTO Employee_Table
VALUES (2, 'Julia Roberts', 987655);

INSERT INTO Employee_Table
VALUES (3, 'Tom Cruise', 987656);
```



Example of insertion on this table response.

PRODUCT TABLE

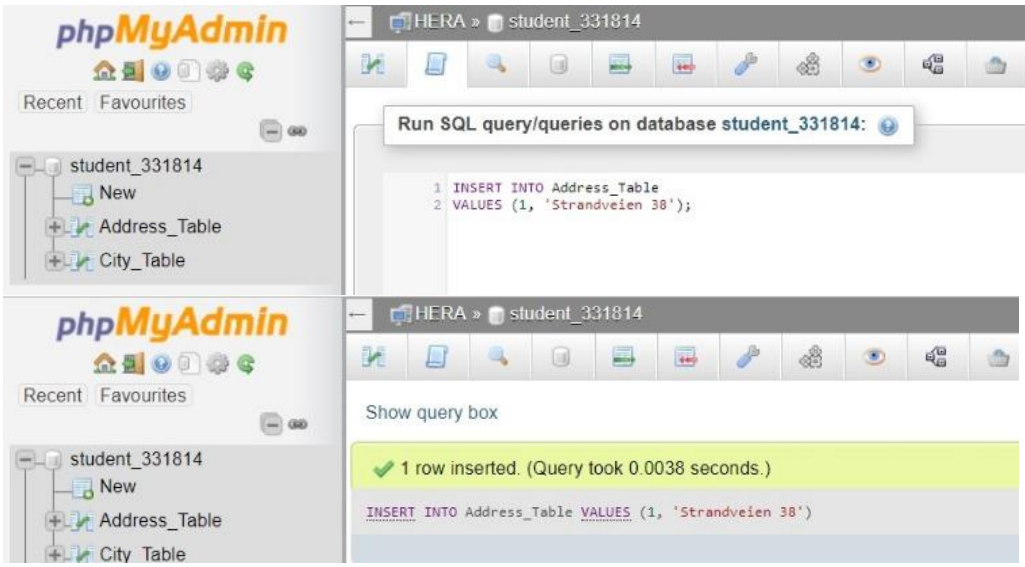
```
/* Create table */
CREATE TABLE Product_Table(
    Product_id INT,
    Descr VARCHAR(255),
    Unit_price INT,
    PRIMARY KEY (Product_id)
);

/* Insert values */
INSERT INTO Product_Table
VALUES (12, 'Pencil', 5);

INSERT INTO Product_Table
VALUES (13, 'Rubber', 2);

INSERT INTO Product_Table
VALUES (14, 'Ruler', 6);

INSERT INTO Product_Table
VALUES (15, 'Sharp', 7);
```



Example of insertion on this table response.

ADRESS TABLE

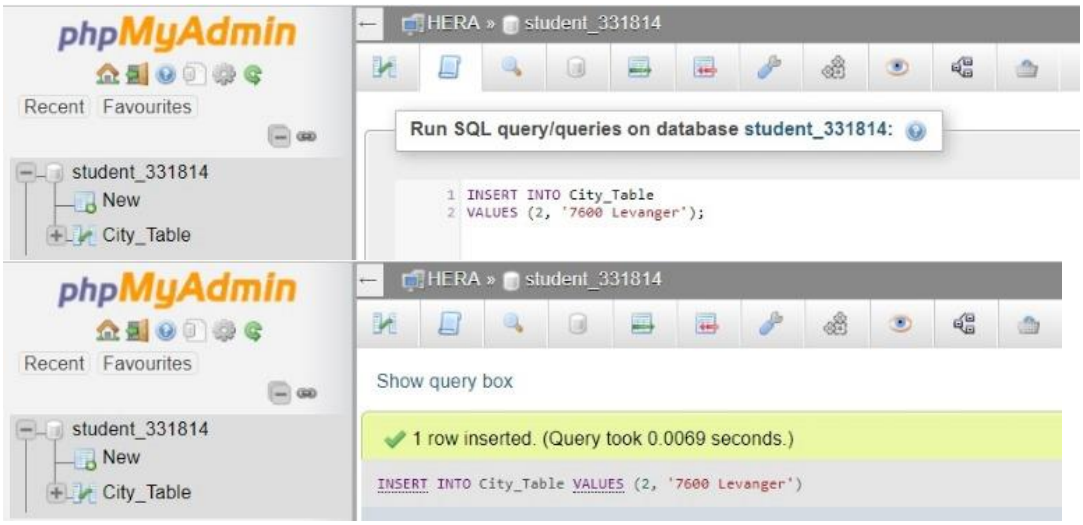
```
/* Create table */
CREATE TABLE Address_Table(
    Address_id INT,
    Cust_addr VARCHAR(255),
    PRIMARY KEY (Address_id)
);

/* Insert values */
INSERT INTO Address_Table
VALUES (1, 'Strandveien 38');

INSERT INTO Address_Table
VALUES (2, 'Kongens 39');

INSERT INTO Address_Table
VALUES (3, 'Hogskolevegen 27');
```



Example of insertion on this table response.

CITY TABLE

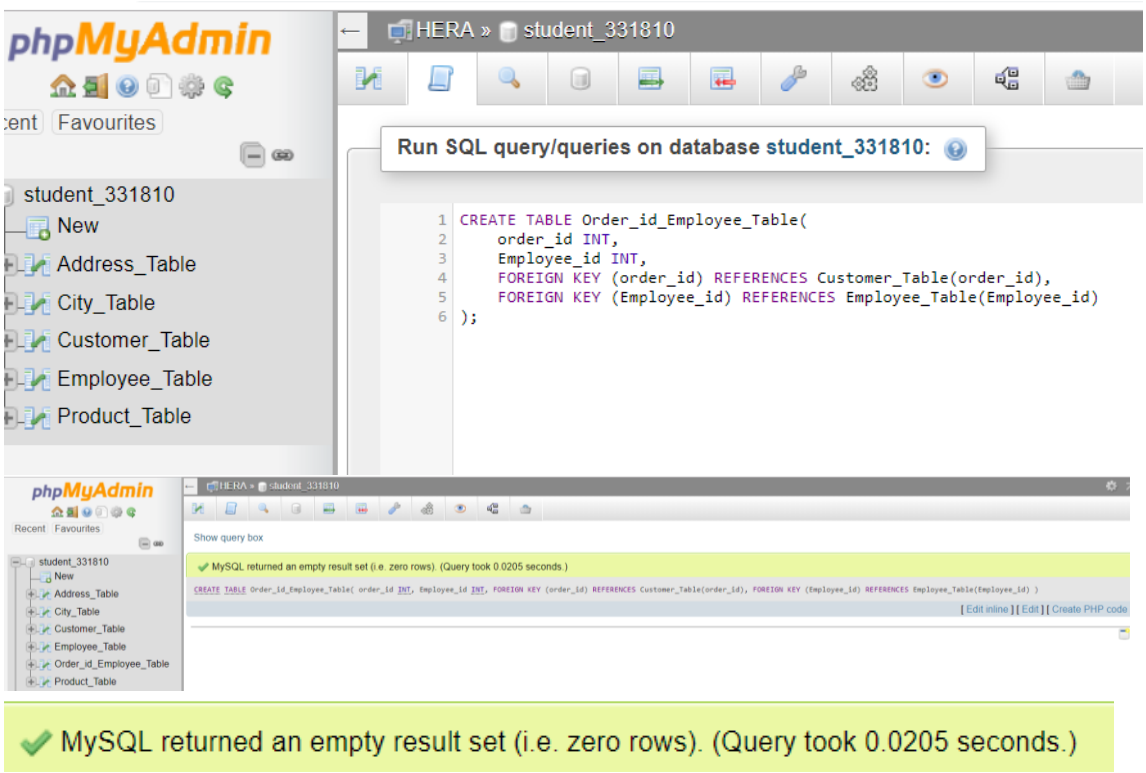
```
/* Create table */
CREATE TABLE City_Table(
    City_id INT,
    City VARCHAR(255),
    PRIMARY KEY (City_id)
);

/* Insert values */
INSERT INTO City_Table
VALUES (1, '7713 Steinkjer');

INSERT INTO City_Table
VALUES (2, '7600 Levanger');
```



Example of insertion on this table response.

ORDER_EMPLOYEE TABLE

```
/* Create table */
CREATE TABLE Order_id_Employee_Table(
    order_id INT,
    Employee_id INT,
    FOREIGN KEY (order_id) REFERENCES Customer_Table(order_id),
    FOREIGN KEY (Employee_id) REFERENCES
Employee_Table(Employee_id)
);
```

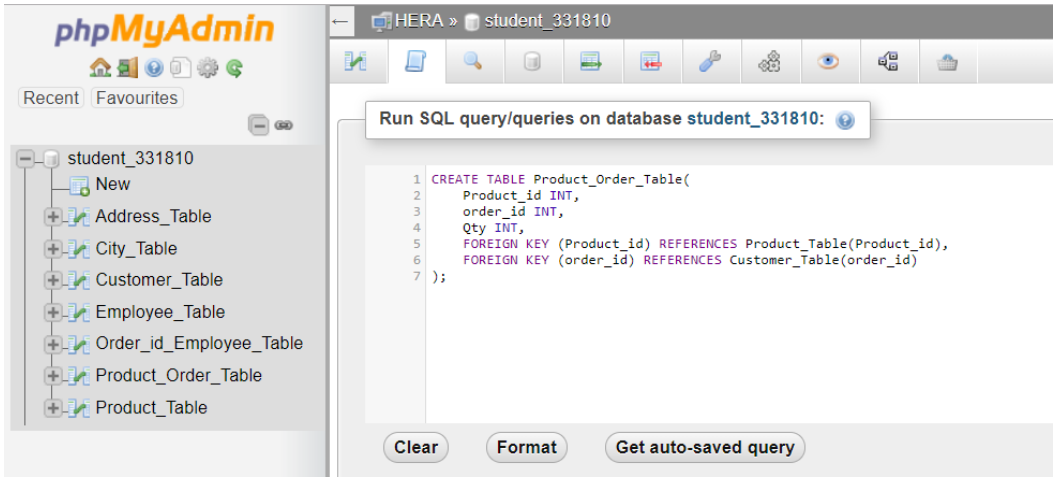


Example of table creation response.

```
/* Insert values */
INSERT INTO Order_id_Employee_Table
VALUES (110, 1);

INSERT INTO Order_id_Employee_Table
VALUES (111, 2);

INSERT INTO Order_id_Employee_Table
VALUES (112, 3);

INSERT INTO Order_id_Employee_Table
VALUES (113, 1);
```

PRODUCT_ORDER TABLE

```
/* Create table */
CREATE TABLE Product_Order_Table(
    Product_id INT,
    order_id INT,
    Qty INT,
    FOREIGN KEY (Product_id) REFERENCES
Product_Table(Product_id),
    FOREIGN KEY (order_id) REFERENCES Customer_Table(order_id)
);
```



Example of table creation response.

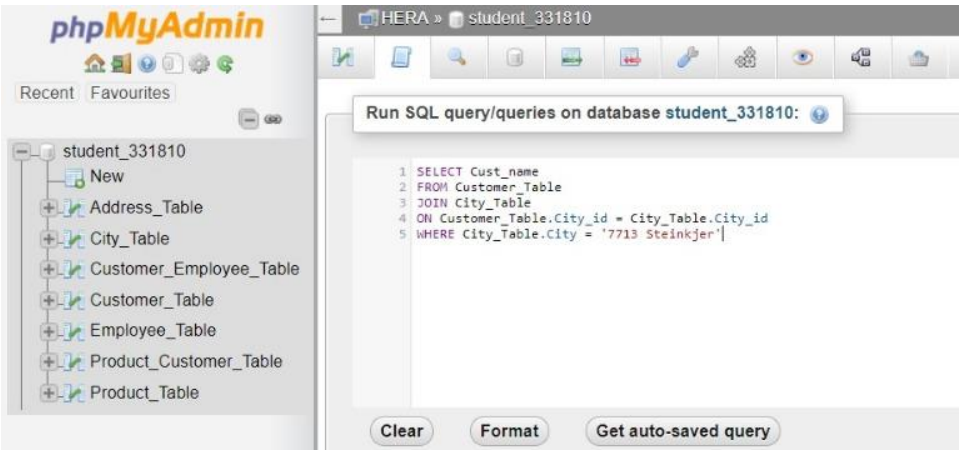
```
/* Insert values */
INSERT INTO Product_Order_Table
VALUES (12, 110, 50);

INSERT INTO Product_Order_Table
VALUES (13, 110, 10);

INSERT INTO Product_Order_Table
VALUES (12, 111, 1);

INSERT INTO Product_Order_Table
VALUES (14, 111, 1);

INSERT INTO Product_Order_Table
VALUES (15, 112, 1);

INSERT INTO Product_Order_Table
VALUES (12, 113, 10);

INSERT INTO Product_Order_Table
VALUES (13, 113, 10);
```

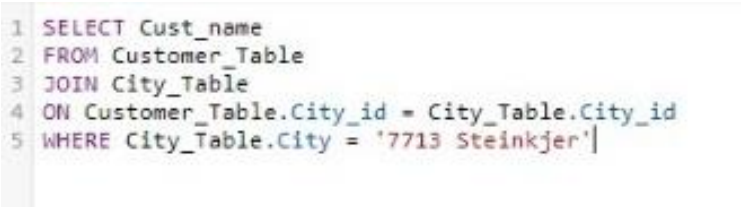
Tasks:

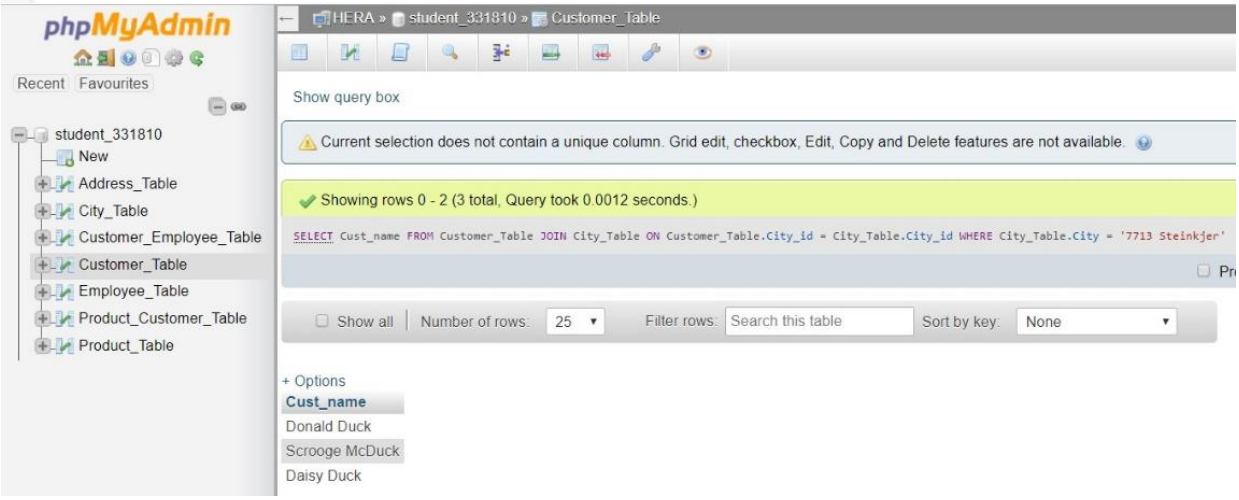
- With SQL-queries we can find some relevant information in an easy way making use of our own code and the data base :

/* a. Find all the customers who live in Steinkjer. */

```
SELECT Cust_name
FROM Customer_Table
JOIN City_Table
ON Customer_Table.City_id = City_Table.City_id
WHERE City_Table.City = '7713 Steinkjer'
```





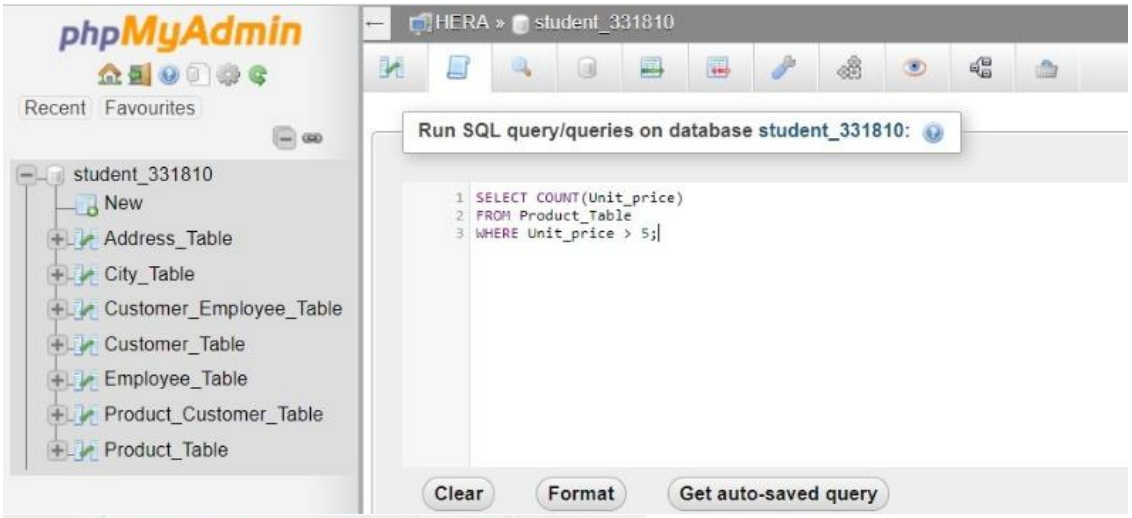


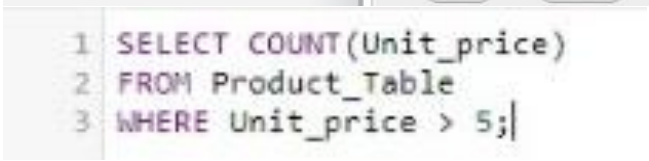
+ Options

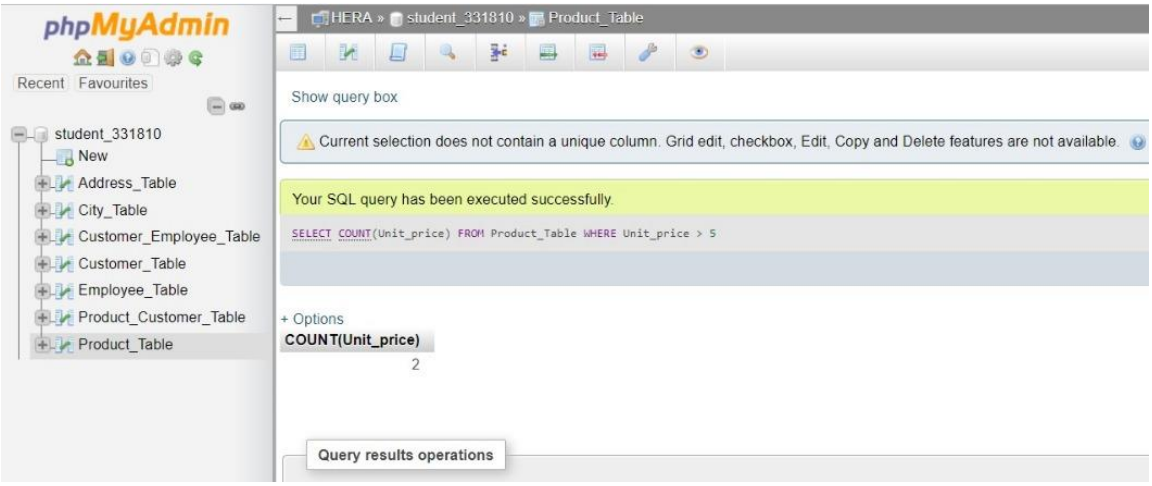
Cust_name
Donald Duck
Scrooge McDuck
Daisy Duck

/* b. Find all the number of products with price per unit larger than 5000. */

```
SELECT COUNT(Unit_price)
FROM Product_Table
WHERE Unit_price > 5;
```





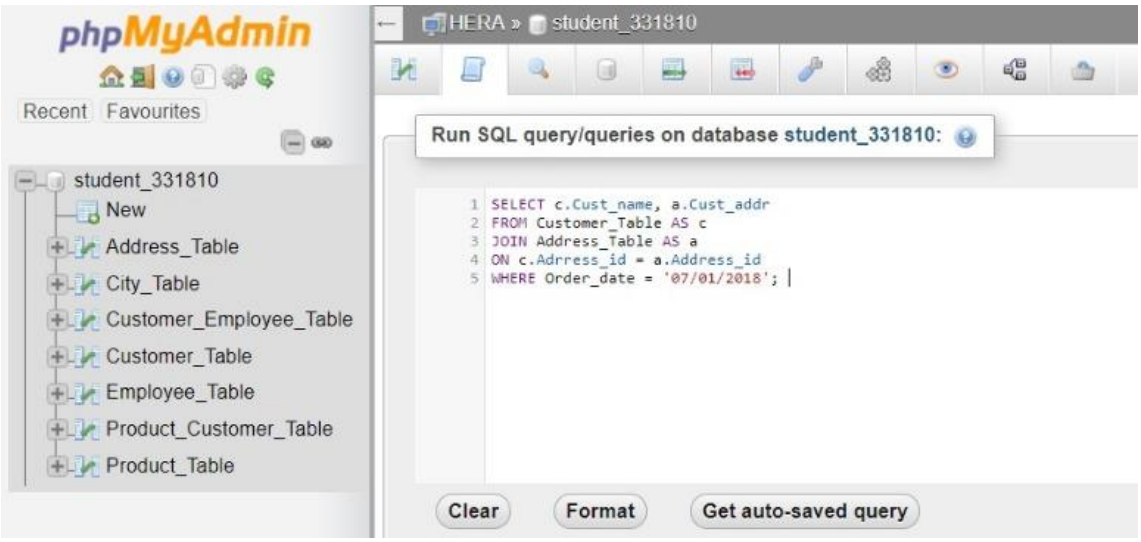


/* c. List the customer name and address for orders placed in June 2018.*/

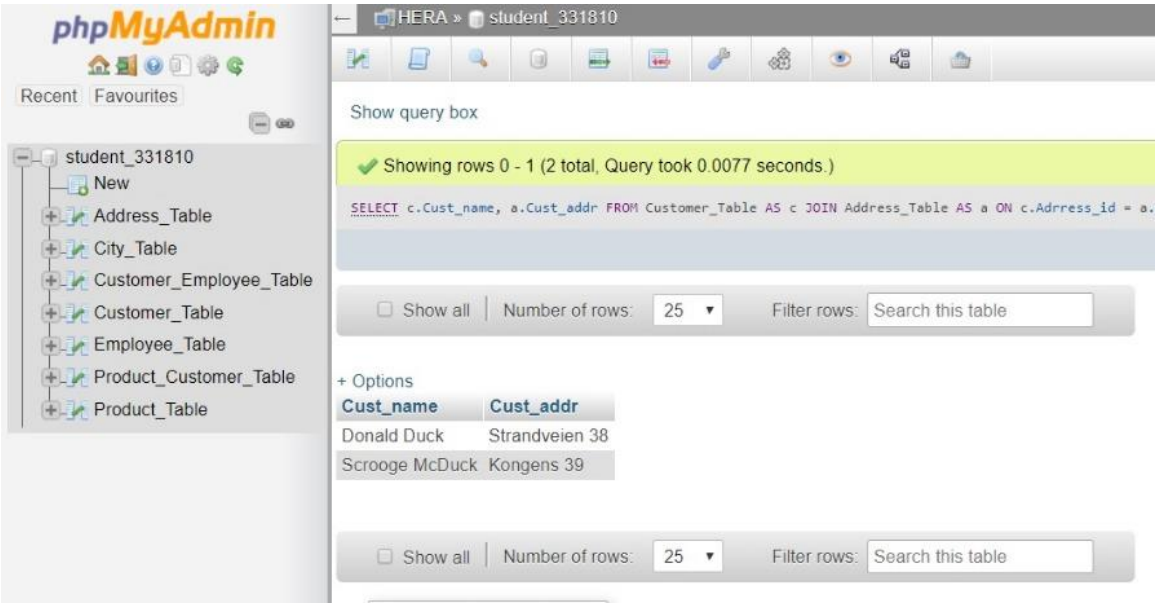
```
SELECT c.Cust_name, a.Cust_addr
FROM Customer_Table AS c
JOIN Address_Table AS a
ON c.Adrress_id = a.Address_id
WHERE Order_date = '07/01/2018';
```

*In this case the data that we supposed to find is not on the data base, when we try to access to this information we get NULL.

In order to show we can find similar data we have change the date of id for one we know it is on the data base (January instead of Jan).




```
SELECT c.Cust_name, a.Cust_addr
FROM Customer_Table AS c
JOIN Address_Table AS a
ON c.Adrress_id = a.Address_id
WHERE Order_date = '07/01/2018';
```



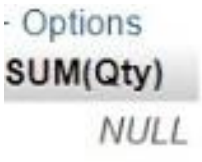
+ Options

Cust_name	Cust_addr
Donald Duck	Strandveien 38
Scrooge McDuck	Kongens 39

/* d. Find the name of the employee(s) responsible for the sales to Scrooge McDuck in May 2018. */

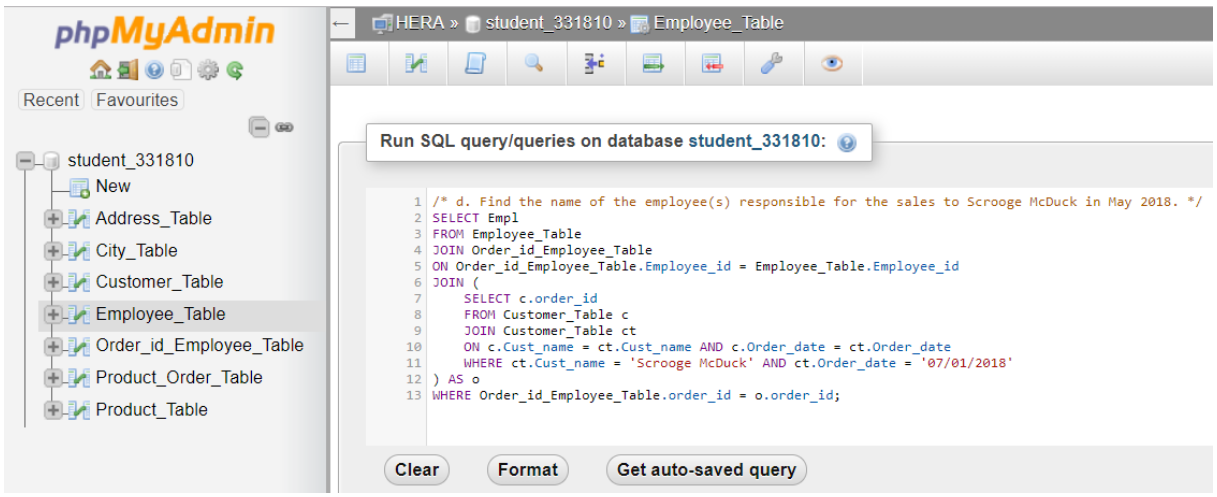
```
SELECT Empl
FROM Employee_Table
JOIN Order_id_Employee_Table
ON Order_id_Employee_Table.Employee_id =
Employee_Table.Employee_id
JOIN (
    SELECT c.order_id
    FROM Customer_Table c
    JOIN Customer_Table ct
    ON c.Cust_name = ct.Cust_name AND c.Order_date =
ct.Order_date
    WHERE ct.Cust_name = 'Scrooge McDuck' AND ct.Order_date =
'07/01/2018'
) AS o
WHERE Order_id_Employee_Table.order_id = o.order_id;
```

*In this case the data that we supposed to find is not on the data base, when we try to access to this information we get NULL.

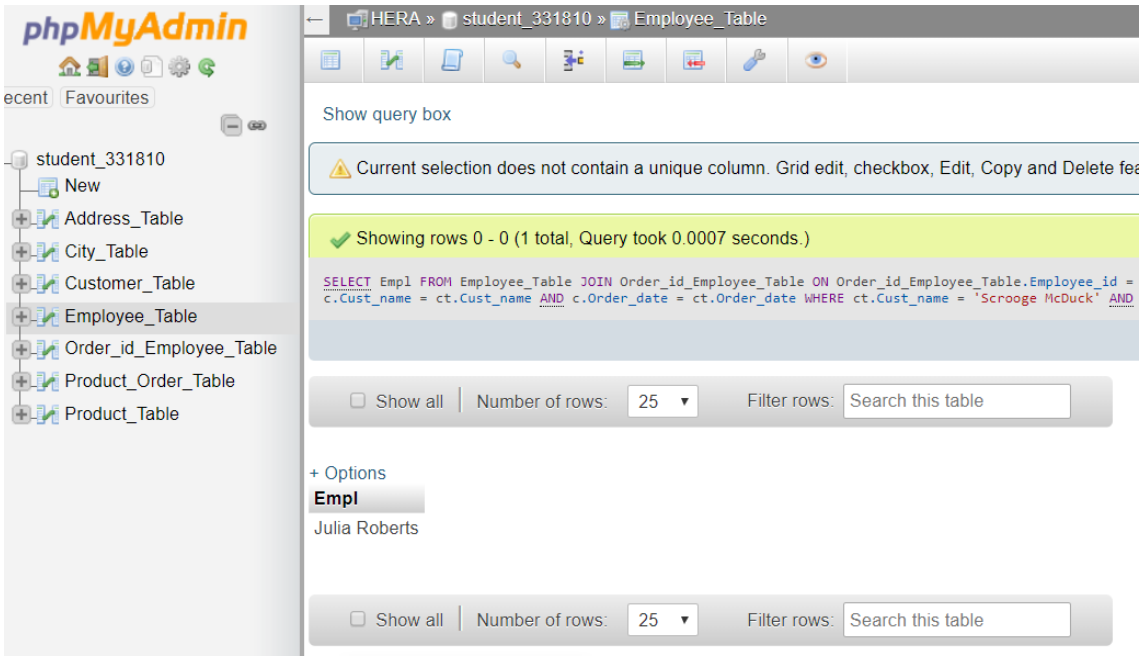


In this case the result is null because there are no value for the requirements on the selected dates for the selected customer.

In order to show we can find similar data we have change the date for one we know it is on the data base (January 1 instead of May).



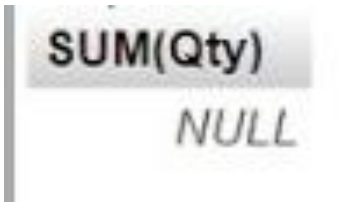
```sql
/* d. Find the name of the employee(s) responsible for the sales to Scrooge McDuck in May 2018. */
SELECT Empl
FROM Employee_Table
JOIN Order_id_Employee_Table
ON Order_id_Employee_Table.Employee_id = Employee_Table.Employee_id
JOIN (
    SELECT c.order_id
    FROM Customer_Table c
    JOIN Customer_Table ct
    ON c.Cust_name = ct.Cust_name AND c.Order_date = ct.Order_date
    WHERE ct.Cust_name = 'Scrooge McDuck' AND ct.Order_date = '07/01/2018'
) AS o
WHERE Order_id_Employee_Table.order_id = o.order_id;
```



/* e. Find the total amount for order with Order_id=117. */

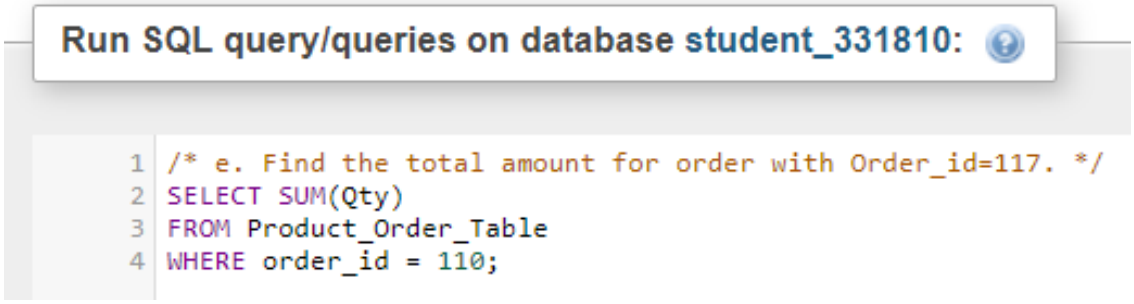
```sql
SELECT SUM(Qty)
FROM Product_Order_Table
WHERE order_id = 110;
```

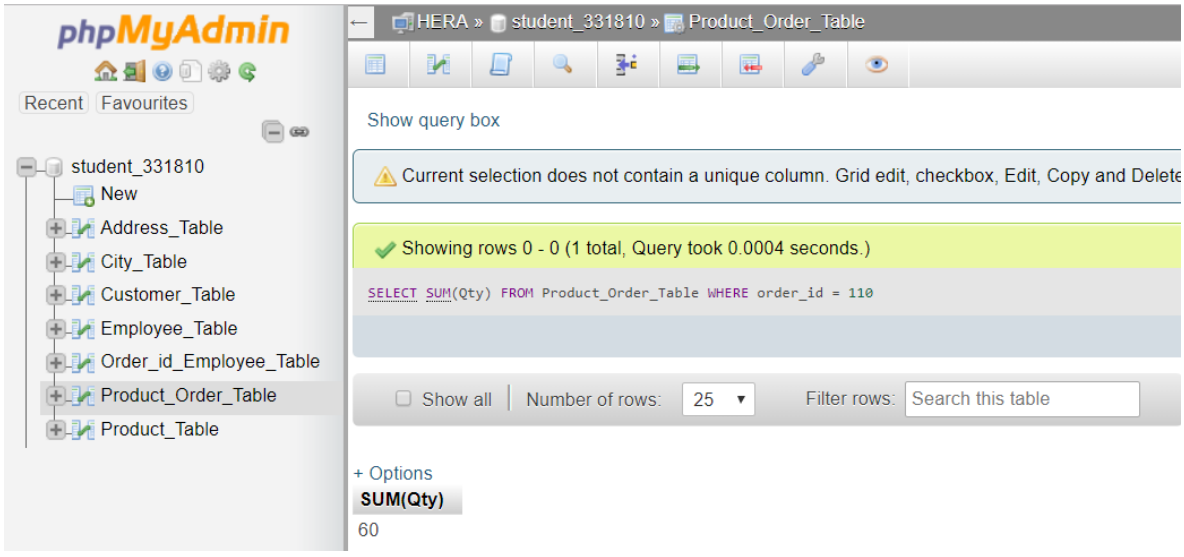
*In this case the data that we supposed to find is not on the data base, when we try to access to this information we get NULL.



*In this case we obtain null because the value we supposed to obtain is a integer.

In order to show we can find similar data we have change the number of id for one we know it is on the data base (110 instead of 117).





+ Options

SUM(Qty)
60
