Project Report

Author: Stavri Gkousi

Subject: CS 312 Database Management Systems

My fashion design company

As the topic of this project I have chosen to upgrade my fashion company used in the previous classes project. This fashion company is worldwide operational and it’s goal is to periodically prepare Summer and Winter collections, get orders and ship the goods to the customers.

This company has 40 freelancer employees, 6 collections, 12 products, 40 customers and has gotten 80 orders totalling at 150 items.

The freelancers are appointed to only one assignment. It shows that the company only hires people when needed on a project thus their payment is based on the assignments they partake and not on a monthly basis.

Each customer can have one or many orders and each order contains a multitude of products(items).The shipping process happens in two parts.

The inspiration for this project is the fact that for a long time I have been very interested in the fashion industry and with this project I have been able to gain an insight into how these companies work in real life. The database that I have developed can be very useful for people who occupy the position of CEO, directors, accountants or department managers.

A good use of these database is the fact that it can be very useful for automating systems like hiring policies with the help of trigger queries.

In this project we have a M-N relationship between the table’s orders and items. It contains a 1-1 relationship and a hierarchical relationship in the employee’s table. The hierarchical relationship is between the employee\_id and managers. The queries I have made can be very useful in a real-life situation. The queries range in relation to the lessons we have done in class, they contain subqueries(correlated and non-correlated), hierarchical queries, PL/SQL procedures, functions and triggers. The queries can be used to filter, sort and display simple data we need. The PL/SQL queries are used to provide complex information, show database relations and create ways to trap errors so that our database can maintain its integrity.

TABLE DESCRIPTION

Table name: Employees

A screenshot of a computer

Description automatically generated

Employee\_ID: shows a unique text that depicts each employee

Employee\_Title: The title of the person depending on their gender(used to generate male or female names)

Employee\_name: Displays the name of the employee

Employee\_surname: Displays the surname of the employee

BirthDate: Shows the birthdays of the employees

Address: Displays the address of the employees

Phonenumber: Displays a form to contact the employees in this case a phone number

Bonus: a percentage value to be added to the payment they get from the project. This bonus isn’t specific because it is a negotiable value

Manager: Contains all the IDs of the managers of the company

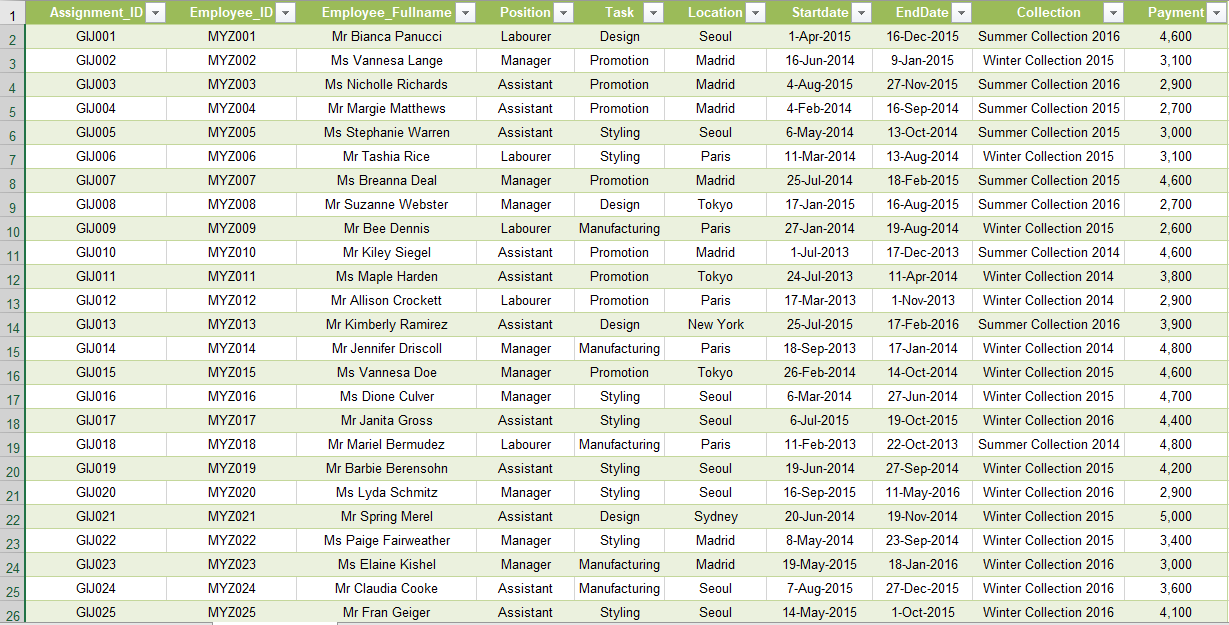


Table name: Assignments

Assignment\_ID: a unique text to identify the assignments

Employee\_ID: shows a unique text that depicts each employee to be used as a foreign key

Employee\_Fullname: Displays the full name of the employees with their title

Position: The position an employee has on a project, this position can be Manager,Assistant, Labourer

Task: Shows the task the employee has worked on. This task can be Design, Styling, Manufacturing, and Promotion.

Location: Shows where the employee is currently working. Multiple employees can work on a project yet be in different cities in the world

Startdate: Displays the date when work started on a project. The data used is random but it is limited to a few months in the year because the company makes 2 collections in a year

EndDate: Displays the date when work on the project is finished. The date is randomized to always be bigger than the starting date yet it finished a few months before the collection is said to be released.

Collection: The project each employee was assigned to. The collection is created in the years 2014, 2015, 2016. The collection is made of the Summer and Winter collections.

Payment: The employee payment for the project. Payments aren’t attached to a position because workers might be in different locations.

A screenshot of a computer

Description automatically generated

Table name: Customers

Customer\_ID: unique text to identify each customer

Title: the title related to the customer's gender

Firstname: The first name of the customer

Lastname: The last name of the customer

BirthDate: The birthdate of all the customers

Address: Displays the address of the customers

Phonenumber: Displays customer's way of communication in this case a phone number

Area: Displays the continent where a customer is located

Rank: A form of ranking priority-wise of customers. The ranks are Platinum, Gold, Silver, Bronze

Preferred\_Paymnet: The preferred way a customer likesto pay. The types are Cash, Credit Card, Bank Wire

A screenshot of a computer

Description automatically generated

Table name: Products

Producst\_ID: Unique text to identify a product

Name: The name of the product

Collection: The collection each employee was assigned to

Cost: The cost to make a product

Price:The price of a product

A screenshot of a data sheet

Description automatically generated

Table name: Orders

Order\_ID: Unique text to identify orders

Customere\_ID: unique text to identify each customer and used as a foreign key

Order\_Date: The date when the order was made

Pay\_Type: The method with which the payment was made

Order\_Status: The status of the order

Cost\_FirstShipment: How much the first shipment cost

Cost\_SecondShipment: How much the second shipment cost

A screenshot of a table

Description automatically generated

Table name: Items

Order\_ID: Unique text to identify orders and used as a foreign key

Product\_ID: unique text to identify each product and used as a foreign key

Total\_Quantity: Shows the total quantity of products in an order

Quantity\_FirstShipment: The quantity of products in the first shipment

Quantity\_SecondShipment: The quantity of products in the second shipment

Queries

Querry1: Display the quantity of winter products during the month of January in descending order

select p.product\_id product\_id ,p.name name,sum(i.total\_quantity) total\_quantity

from products p,items i,orders o

where p.product\_id=i.product\_id and o.order\_id=i.order\_id and p.collection like 'Winter%'

and to\_char(o.order\_date,'MM')='01'

group by p.product\_id, p.name

order by sum(i.total\_quantity) desc

A screenshot of a computer

Description automatically generated

Querry 2: Display the employee's full name, birthdate, position, and collection they work on

select a.employee\_fullname,e.birth\_date,a.position,a.collection

from employees e,assignments a

where e.employee\_id=a.employee\_id

order by e.birth\_date

A screenshot of a computer

Description automatically generated

Querry 3: Display the most populated positions

select count(assignment\_id),task

from assignments

group by task

order by 1

A black and grey striped background

Description automatically generated

Querry 4: Display the shipped orders where customers paid with their preferred payment type

select o.order\_id order\_ID, c.preferred\_payment prefered\_payment, o.pay\_type payment\_type

, o.order\_status Order\_status

from orders o join customers c

on o.customer\_id = c.customer\_id

and c.preferred\_payment = o.pay\_type

and o.order\_status = 'Shipped'

order by o.order\_id;

A screenshot of a computer

Description automatically generated

Querry 5: Display info for the customer who had the most amount shipped on the first shipment

create or replace view private as

select quantity\_firstshipment quantity,order\_id

from items

order by quantity\_firstshipment desc

select c.\*

from customers c,orders o

where o.customer\_id=c.customer\_id

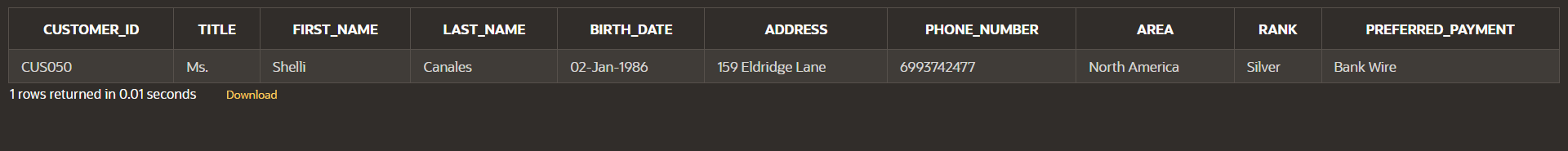
and o.order\_id=(

select order\_id

from private

where rownum=1

)



Querry 6: Display the quantity of all the shipped items, order by date

select i.order\_id||i.product\_id orders,o.order\_date dates,i.total\_quantity

from orders o,items i

where o.order\_id=i.order\_id and o.order\_status='Shipped'

order by o.order\_date

A screenshot of a computer

Description automatically generated

Querry 7: Find the profit from product manufacturing for every customer area and sort in descending order

create or replace view profit\_by\_area as

select c.area "Customer Area", sum((p.price-p.cost)\*i.total\_quantity) "Manufacturing Profit"

from customers c join orders o

on c.customer\_id = o.customer\_id

join items i

on o.order\_id = i.order\_id

join products p

on p.product\_id = i.product\_id

group by c.area

order by c.area desc

select \*

from profit\_by\_area

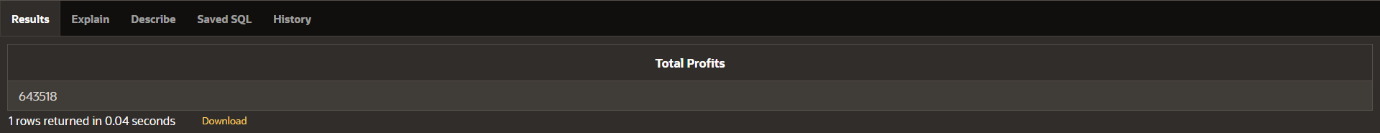
A screenshot of a computer

Description automatically generated

Querry 8: Find total profit from products

select sum("Manufacturing Profit") "Total Profits"

from profit\_by\_area;



Querry 9: Find the status of orders where the first shipment costs more than the second one, while it contains less quantity of products

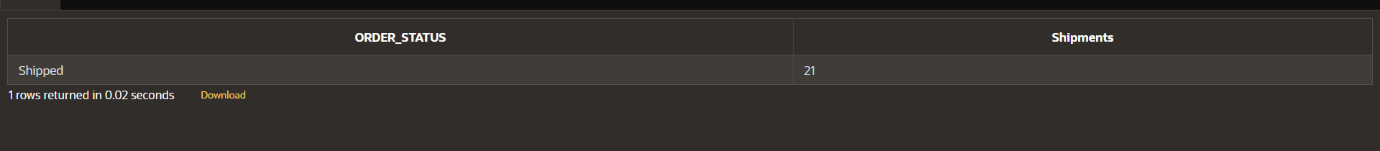
select o.order\_status,count(i.order\_id) "Shipments"

from orders o,items i

where i.order\_id=o.order\_id and i.quantity\_firstshipment<i.quantity\_secondshipment

and o.cost\_firstshipment>o.cost\_secondshipment and o.order\_status='Shipped'

group by o.order\_status



Querry 10:Find the profit from product manufacturing for every payment type

create or replace view payment\_profit as

select o.pay\_type "Payment Type", sum((p.price-p.cost)\*i.total\_quantity) "Manufacturing Profit"

from customers c join orders o

on c.customer\_id = o.customer\_id

join items i

on o.order\_id = i.order\_id

join products p

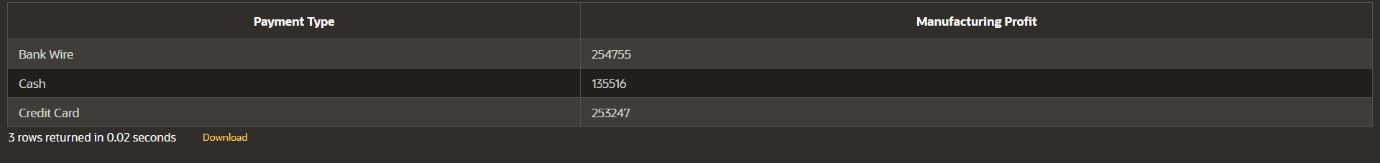
on p.product\_id = i.product\_id

group by o.pay\_type

order by o.pay\_type;

select \*

from payment\_profit;



Querry 11: Show the quantity of every product ordered

create or replace view view\_1 as

select product\_id, name

from products

create or replace view view\_2 as

select product\_id,sum(total\_quantity) quantity

from items

group by product\_id

order by 1

select sum(decode(a.name,'Shirt',b.quantity)) "Shirt",

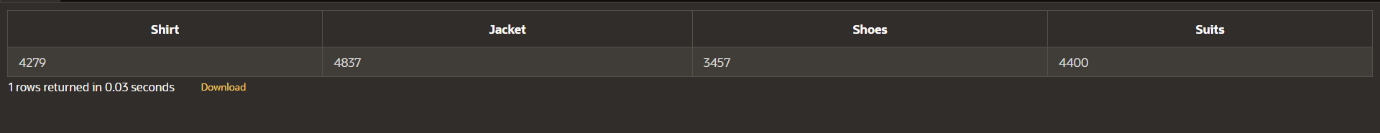
sum(decode(a.name,'Jacket',b.quantity)) "Jacket",

sum(decode(a.name,'Shoes',b.quantity)) "Shoes",

sum(decode(a.name,'Suits',b.quantity)) "Suits"

from view\_1 a,view\_2 b

where a.product\_id=b.product\_id



Querry 12 Employee with id: MYZ040 has left the company,delete useless record

create or replace procedure deletion(id\_del in varchar2)

is

begin

delete from assignments

where employee\_id=id\_del;

delete from employees

where employee\_id=id\_del;

end deletion;

begin

deletion('MYZ040');

end;



Querry 13 check condition if what we insert into employees is null

create or replace trigger emp\_check\_null

before insert on employees

for each row

declare

v\_check\_null number;

begin

if :new.employee\_title is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.employee\_name is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.employee\_surname is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.birth\_date is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.address is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.phone\_number is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.bonus is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

elsif :new.managers is null then

raise\_application\_error(-20003,'Null value detected,prohibited on this table');

end if;

end;

A black rectangular object with white text

Description automatically generated

Querry 14 find the employee with the second lowest salary in the company working on a summer collection

create or replace view is\_summer as

(

select s.\*

from assignments s

where s.collection like 'Summer%'

)

select s.employee\_id,s.employee\_fullname,s.payment

from is\_summer s

where 2=

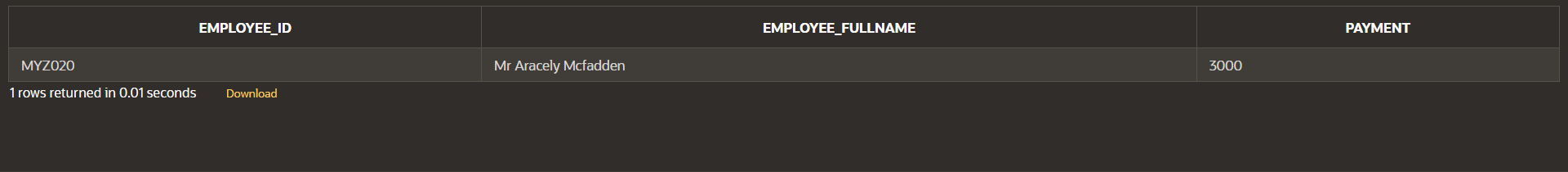
(

select count(distinct(s2.payment))

from is\_summer s2

where s2.payment<=s.payment

)



Querry 15 find who is the top excecutive in the company and who are his employees

declare

number\_of\_emp number;

v\_record employees%rowtype;

cursor top\_manager is

select count(employee\_id)

from employees

where managers =

(

select employee\_id

from employees

where managers is null

);

begin

open top\_manager;

loop

fetch top\_manager into number\_of\_emp;

exit when top\_manager%notfound;

select \* into v\_record

from employees

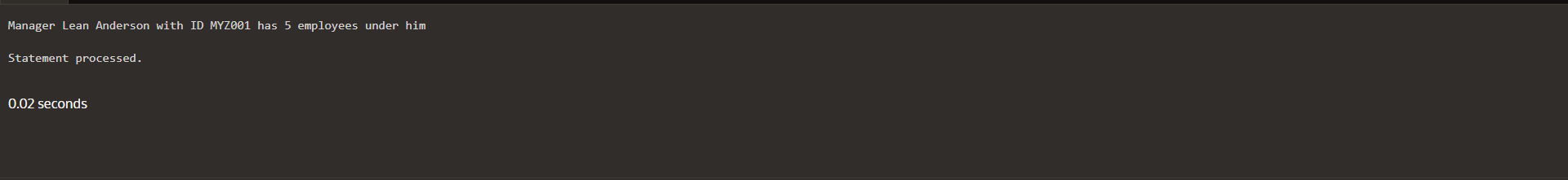
where managers is null;

DBMS\_OUTPUT.PUT\_LINE('Manager '||v\_record.employee\_name||' '||v\_record.employee\_surname||' with ID '||v\_record.employee\_id||' has '||number\_of\_emp||' employees under him');

end loop;

close top\_manager;

end;



Querry 16 Display a report of employee id, employee full name and location for all employees who earn more than the average salary.

create or replace view salaries as (

select a.employee\_id emp\_id, a.employee\_fullname fullname, a.location location, a.payment + (a.payment\*(e.bonus/100)) salary

from assignments a

join employees e

on (a.employee\_id = e.employee\_id)

);

select \*

from salaries

where salary >

(

select avg(salary)

from salaries

);



Querry 17. Modify the previous query to display information for employees who worked in assignments located in Paris.

select \*

from salaries

where salary >

(

select avg(salary)

from salaries

)

and location in

(

select location

from salaries

where location = 'Paris'

);

A screenshot of a computer

Description automatically generated

Querry 18 Display Managers hierarchy

select level, lpad(' ',LEVEL+1, '$') || employee\_surname "Employee",

employee\_id, managers

from employees

start with managers is null

connect by prior employee\_id = managers

order siblings by employee\_surname;

A screenshot of a computer

Description automatically generated

Querry 19 Show every relation between all employees in a path format.

select sys\_connect\_by\_path(employee\_id||' '||employee\_surname, '=>') Relations

from employees

start with managers is null

connect by prior employee\_id = managers;

A screenshot of a computer

Description automatically generated

Querry 20 Show employees who aren't managers

select employee\_id||' '||employee\_surname "Employees who are not managers"

from employees

where connect\_by\_isleaf=1

start with managers is null

connect by prior employee\_id=managers;

A screenshot of a computer

Description automatically generated

Querry 21 Calculate the profit per unit and display this information in the format:

"Product Name with ID:XXX of Collection name gives a profit of x% per unit"

declare

cursor find is

select \*

from products;

profit number;

v\_record products%rowtype;

begin

open find;

loop

fetch find into v\_record;

profit :=v\_record.price-v\_record.cost;

exit when find%notfound;

dbms\_output.put\_line('Product '|| v\_record.name ||' with ID: '|| v\_record.product\_id || ' of collection: ' || v\_record.collection ||' gives a profit of '|| profit || '$ per unit.');

end loop;

close find;

end;

A screenshot of a computer

Description automatically generated

Querry 22 create a trigger which logs error codes and when they happened for orders table

create table error\_log(

error\_code number(6) not null,

error\_message varchar2(4000),

date\_of date not null

);

select \*

from error\_log

create or replace trigger error\_catch

after insert on orders

for each row

declare

error\_code number(6);

error\_message varchar2(4000);

date\_of date:=sysdate;

null\_values exception;

prime\_key exception;

begin

error\_code:=sqlcode;

error\_message:='Error: '||sqlerrm;

if :new.customer\_id is null or :new.order\_date is null or :new.pay\_type is null or :new.order\_status is null or:new.cost\_firstshipment is null or :new.cost\_secondshipment is null then

raise null\_values;

end if;

exception

when null\_values then

insert into error\_log values(-20034,'Error: Null values not allowed',date\_of);

dbms\_output.put\_line('Error: Null values not allowed');

when others then

insert into error\_log values(error\_code,error\_message,date\_of);

dbms\_output.put\_line(error\_message);

end;

-- to test

insert into orders values ('OID231', 'CUS001', '31-Dec-2018', 'Bank Wire', 'Ready to ship', 482, null);

A screenshot of a computer

Description automatically generated

Querry 23 Calculate projected total costs

create or replace function shipping\_costs

return number

is

v\_first number :=0;

v\_second number :=0;

v\_cost number;

cursor my\_cur is

select sum(cost\_firstshipment), sum(cost\_secondshipment)

from orders;

begin

open my\_cur;

fetch my\_cur into v\_first,v\_second;

close my\_cur;

v\_cost:=v\_first +v\_second;

return v\_cost;

end shipping\_costs;

create or replace function prod\_cost

return number

is

costs number;

cursor prod\_cur is

select sum(p.cost\*i.total\_quantity)

from products p join items i

using (product\_id);

begin

open prod\_cur;

fetch prod\_cur into costs;

close prod\_cur;

return costs;

end prod\_cost;

create or replace function salary\_cost

return number

is

costs number;

cursor sale\_cur is

select sum(salary)

from salaries;

begin

open sale\_cur;

fetch sale\_cur into costs;

close sale\_cur;

return costs;

end salary\_cost;

-- final display

declare

expenses number;

begin

expenses := shipping\_costs + prod\_cost + salary\_cost;

dbms\_output.put\_line('The total projected expenses of the company in salaries are ' || salary\_cost || '$.');

dbms\_output.put\_line('The total projected expenses of the company in shipping are ' || shipping\_costs || '$.');

dbms\_output.put\_line('The total projected expenses of the company in production are ' || prod\_cost || '$.');

dbms\_output.put\_line('The total projected expenses of the company in production, shipping and salaries are ' || expenses || '$.');

end;

A screen shot of a computer

Description automatically generated

Further development suggestions

This application isn’t in its perfect state and some changes are :

1. To change the relationships between employees and assignments. This would mean that the 1-1 relationship that employees and assignments had would change to a M-N relationship. For this change to take place the company would need to update its freelancing hiring policies.
2. Some additional could be added to the tables, for example the expenses part of the business could also calculate the expenses that come from the materials used, a separate table that would go more into details about the payments of the employees.

The addition of these new data would make the database more complex, but the introduction to functions, procedure, triggers and censors can help us better analyze and work with this new data. Triggers are in particular very useful to us because with them we can implement the actual policies on what form the data must be added to the tables eliminating in this way human errors.

The user input functions are also a very good addition to the database because it can help us eventually create a database where we can create websites that use the information in these databases just like big stores and companies do in real life.