جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⴰⵢⵏ

AL AKHAWAYN UNIVERSITY

Fall 2021

Friday, October 29, 2021

CSC 332 – Database systems

« WeHelpYou »

Soufiane Tounsi

Salma Louhaychi

Ziad El Ismaili

Hamza Zaher

Supervised by:

Pr. Lamiae Bouanane

Table of Content

I. Introduction:

a) Project description:

Many small businesses such as shops, retails stores and hardware stores lack databases in their line of work; in fact, owners usually operate by hand, doing calculations rather than scanning items in a specific device, which is clearly not practical since implementing a database for instance, would clearly help owners by organizing information about the shop, making it easier to access, manage and update the items. It could also help with transactions and product inventory, as well as improving the overall customer satisfaction since fast and easy service will be provided.

However, one of the reasons owners would not implement a database in their commerce might be the lack of interest and knowledge about it, but most importantly, they seem to have a small future vision of their business.

Therefore, today we decided in our project to design an application that will help a small hardware store business. The goal would be facilitating storing and accessing items tasks. The owner could later buy and use advanced devices and gadgets at the checkout, so that items will be scanned and automatically updated on the hardware store database.

b) Client's information:

Client Name: Mr. Abdesselam Rachda.

Hardware Store name: "Aaouinati"

Address: Saada Lotissement 69, Hay Caharf, 40100.

City: Marrakech.

Contact number: (+212) 6 35 73 88 94

c) Project title and team members:

For this project, our team decided to choose the following title: ***WeHelpYou***, and you can find in the table below our team information:

Full Name	Major	Academic Status	E-mail
Ziad El Ismaili	Computer Science	Junior	Z.Elismaili@aui.ma
Soufiane Tounsi	Computer Science	Junior	So.Tounsi@aui.ma
Salma Louhaychi	Computer Science	Junior	S.Louhaychi@aui.ma
Hamza Zaher	Computer Science	Junior	H.Zaher@aui.ma

II. Requirements Gathering:

a) Client's Hardware Store Current System:

For the moment, small business such as shops, retail stores and especially hardware stores, are used to operating on paper using physical file systems to manage their inventory, keep track of sales and doing the necessary calculations to assess their financial growth. In addition, there is no platform that allows customers to keep up with the price, location, or even name of these small shops online. Small business in Morocco, in particular hardware stores, are in desperate need of digitalization of the data they work with.

Therefore, in our case, our client's store is on desperate need to a digitalized platform and a database instead of a physical file, in order to manage well its inventory and make remarkable profit.

b) Importance of a digitalized hardware store:

Today, many independent electricians, plumbers or builders in Morocco pay for their own equipment during a job, which they include in the client bill at the end. Since there is a little number of big surface stores such as "Bricoma" and Ikea that offer off the shelf equipment, the workers are obliged to go to different hardware stores, where they may or may not find the specific parts they are looking for.

Thus, part of our project will focus on providing an online platform to check the availability of products in the hardware stores near you.

C) Objectives and functionalities:

Because of the current problems mentioned above that small non-digitalized hardware stores face, we decided in our project to help our client overcome them, and that is mainly by achieving the following objectives below:

- Create a database that includes information about the hardware store items, their description and price along with any other data appearing in the physical file.
- Use an appropriate DBMS to facilitate operations such as data retrieval and calculations using aggregate functions.
- Data availability, meaning the system should guarantee easy access the data (items), in a meaningful format.
- Provide an online platform in order to check the availability of products.
- Avoid data redundancy and respect all the database criteria. (Data integrity, security, independence…)
- Minimize data inconsistency.

In fact, implementing the objectives above would certainly:

- Overcome almost all the physical file system problems (current system), which will save the client time and resources.
- Facilitate access to our client's hardware store, whose locations are mostly known to the residents of that specific area.
- Centralize all information regarding the location, availability, and prices of products that the hardware store has to offer.

III. Requirements Specification:

a) Functional Requirements:

Concerning the functional requirements part of our project and after talking with our client, we agreed and decided first that functions are divided into 2 categories: ***Admin requirements*** and ***User requirements.***

Admin requirements: Only the manager (which is our client) can access and use the admin functions, such as Manage Items or Update Items…

User requirements*:* functions that could be used by a cashier at the checkout for instance, to compute the total, generate ticket…

N.B: Please note that an admin could also access user functions and requirements; therefore, the manager can access any requirement or functions he wishes.

- ***Manage Items: (As an admin)***

  → *Consisting of SCRUD functions that represent sub-processes such as add_item, remove_item, search_item…*

- ***Update Items: (As an admin)***

  → *The client / manager could easily update the price, quantity, or description of the items, through sub-functions such as edit_item…*

- ***View Items: (As a user & admin)***

  → *Display all items or some of them in tables depending on a given criteria / condition.*

In a detailed view of our functional requirements, we would design our database requirements as the following:

1. Manage Items

1.1 add_item,

Input: item_code, item_name, description, quantity, and all other attributes.

Output: item along with all its attribute added to the database.

Logic used: inserting a row in the table of items existing in the database using SQL

1.2 remove_item,

Input: item_name or item_code (PK) inputed.

Output: item removed to the database.

Logic used: removing a row in the table of items existing in the database, using SQL.

2. Update Items

2.1 update_name,

Input: new item_name entred.

Output: item_name updated in the item table.

Logic used: updating the item_name in the item table using SQL.

2.2 update_quantity,

Input: new quantity entred manually.

Output: item_quantity updated in the item table.

Logic used: updating the item_quantity in the item table using SQL.

2.3 update_description,

Input: new description entered.

Output: description updated in the item table.

Logic used: updating the description in the item table using SQL.

N.B: The item_code cannot not be updated since it is a primary key in the item table and every item has a unique code.

3. View Items

1.1 display_all_items

Input: no input.

Output: table of items.

Logic used: displaying all rows in the table of items existing in the database, using SQL.

1.2 search_item,

Input: item name or code (PK) inputted, or any other attribute the admin would want, under a certain condition.

Output: item / items displayed to the user in a format of tables

Logic used: displaying specific rows in the table of items existing in the database using SQL.

1.3 check_availability,

Input: item name or code (PK) inputted, or any other attribute the admin would want, under a certain condition.

Output: item_quantity displayed in one row intersecting with a column.

Logic used: displaying specific item_quantity of a chosen item database using SQL.

b) Non-Functional Requirements:

For this part, after a long discussion with of our team members we can agreed that certain non-functional requirements would be:

- The Client insist on having a **Reliable Database System.**

- Use an easy **English Graphical Interface** for the users and the client to work with. (After a working English graphical interface, our team could easily translate it to a French / Arabic one).
- Managing time and **respect all the deadlines** provided by the instructor, or even the client.
- **Meet virtually with the client** whenever necessitated and communicate to him the progress of our work.

Moreover, we could think of other ethical and legal non-functional requirements such as:

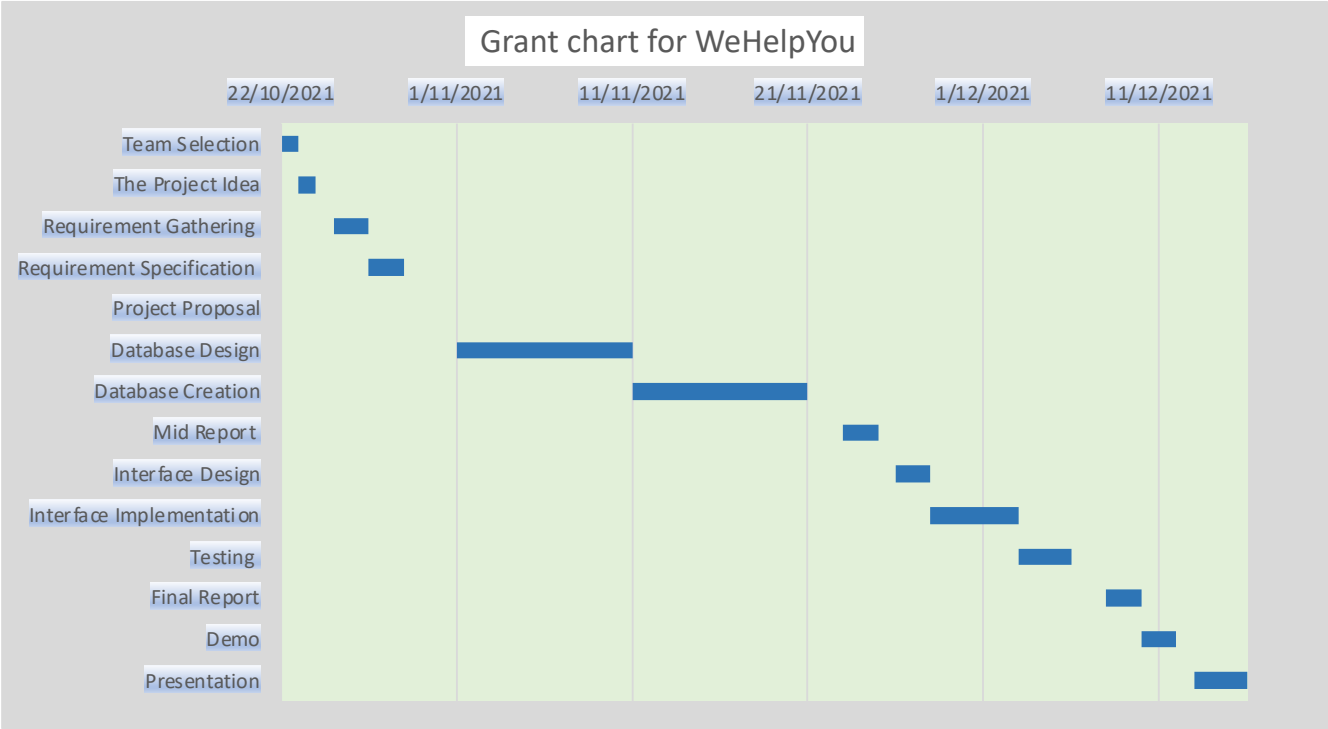
- The system should keep track of the contents entered, making sure that no misleading, unethical or immoral contents is inserted in the database system. Such behavior should not be tolerated.
- Information and data should be secure, protected and preserved within the system. The digitalized system should not share any restricted information unless authorized.

IV. Project Management Plan:

a) Time Management:

To efficiently accomplish this project, a follow up tasks must be attained over time. Those steps are demonstrated each in the table and the Gantt Chart below:

No.Task	Start Date	End Date	Task	Priority	Status
1	22/10/2021	23/10/2021	Team Selection	High	Done
2	23/10/2021	24/10/2021	The Project Idea	High	Done
3	25/10/2021	27/10/2021	Requirement Gathering	Medium	Done
4	27/10/2021	29/10/2021	Requirement Specification	Medium	Done
5	29/10/2021	29/10/2021	Project Proposal	High	Done
6	1/11/2021	11/11/2021	Database Design	High	In Progress
7	11/11/2021	21/11/2021	Database Creation	Medium	To Do
8	23/11/2021	25/11/2021	Mid Report	Low	To Do
9	26/11/2021	28/11/2021	Interface Design	High	In Progress
10	28/11/2021	3/12/2021	Interface Implementation	High	To Do
11	3/12/2021	6/12/2021	Testing	High	To Do
12	8/12/2021	10/12/2021	Final Report	Medium	To Do
13	10/12/2021	12/12/2021	Demo	High	To Do
14	13/12/2021	16/12/2021	Presentation	Medium	In Progress



b) Task distribution:

Concerning the tasks, we agreed that we will try as much as possible to get each member interacting with other's tasks, in order to improve the quality of our final project

V. Design:

A) Conceptual design:

1) Requirement's specification: main system processes

Before creating our project's ER model, we need first to highlight and discuss the entities, which some were previously mentioned in the requirement specification part. In short, the entities are the following:

- ***user**, is **manager** or **cashier***
- ***log_register***
- ***item***

- ***supplier***
- ***aouinati_store***
- ***customer***

User entity should be divided to 2 entities, meaning in a database design perspective, we can have the user entity as a superclass, including only shared attributes between **admin** and **cashier**, that are the 2 possible sub-classes of it. **User** entity should contain the following attributes:

USER entity attributes:

- user_ID *int, PK*
- user_password *varchar (20)*
- user_fname *varchar (20)*
- user_lname *varchar (20)*
- user_email *varchar (40)*
- *user_phone varchar (15)*

Consequently, we need to assign different attribute for both user entity sub-classes:

MANAGER entity attributes:

- user_ID *int, PK*

CASHIER entity attributes:

- user_ID *int, PK*
- salary *int*

Moreover, the other main difference between the 2 sub-classes might be the functions that each entity could perform, for instance, admin can update_item(), while the cashier has only few limited functions.

Also, we need a 1-to-1 relationship between **cashier** entity and **log_register** entity, where it contains the different attribute necessary to keep track of the cashier working day. Please keep in mind that since it is a 1-to-1 relationship, we had the choice to simply add the **log_register** attributes to the cashier entity; however, this is not optimal because the **log_register** attributes (working hours, in time, out time…) are subject to change every day, and it would be better if we had a different entity that handles the work of a cashier.

LOG_REGISTER entity attributes:

- user_ID *int, PK*
- *in_time varchar (40)*
- *out_time varchar (40)*
- working_hours *int*
- _date *date*

One of the important entities is certainly the **item** entity, where it contains all information concerning the products available in the store, as well as products **manager** wishes to add to the store.

ITEM entity attributes:

- item_code *int, PK*
- description varchar (40)
- in_date *date*
- qoh *int*
- price *decimal (6.2)*
- discount_rate *decimal (3.2)*

Now, according to the client, each item has a **supplier**, and some might not have one (i.e few ones made by aouinati store). Therefore, we need a **supplier** entity linked to the **item** entity, where we can find information concerning the supplier of any product in the store. (Foreign key supplier_ID is assigned and displayed in the ERD in the next part..)

SUPPLIER entity attributes:

- supplier_ID *int, PK*
- supplier_name *varchar (40)*
- supplier_phone *varchar (15)*
- supplier_address *varchar (40)*
- supplier_country *varchar (10)*
- supplier_city v*archar (10)*

Furthermore, the manager / client insisted on having a discount for some **customer**, through a loyalty card that contains information about each **customer** such as name, address and more importantly the discount.

CUSTOMER entity attributes:

- cus_ID *int, PK*
- cus_fname *varchar (40)*
- cus_lname *varchar (40)*
- cus_address *varchar (40)*
- *cus_city varchar (40)*
- cus_discount *decimal (3,2)*

Item is found in **aouinati_store, customer** is registred in aouinati_store… In fact, another highly important entity is **aouinati_store,** where items are stored. The entity contains also information about the store, since one of the main objectives of this project is to automate the business of the hardware store and make it available to customer on the internet. Therefore, the **aouinati_store** entity should include public information such as address, city, country, phone and a name as a primary key.

AOUINATI_STORE entity attributes:

- store_name *varchar (40), PK*
- store_address *varchar (40)*
- store_country *varchar (15)*

- store_city *varchar (15)*
- *store_phone varchar (15)*

Lastly, in order for a customer to buy items, an invoice should be generated, with the following attributes:

INVOICE entity attributes:

- invoice_num *int, PK*
- invoice_date *date*

N.B: Foreign keys, relationships, constraints, and other features will be displayed in the next part when designing the ER model.

2) Business rules, entities and relationships:

Some of the business rules were already stated above when describing the use of entities; however, in this part we will highlight and focus more on business rules that will be converted to relationships, following the client demands:

Relationships:

- ***user is manager:*** "a user can be the manager".
- ***user is cashier:*** "a user can be a cashier".
- ***cashier has log_register:*** "one cashier has only many log registers, one for each day but a log_register contains information about only one cashier".

→ ***cashier 1:M log_register***

- ***cashier works in aouinati_store:*** "one cashier works in aouinati_store but aouinati_store can have multiple cashiers work inside as cashiers".

→ ***aounati_store*** *1:M* ***cashier***

- ***supplier supplies item:*** "a supplier supplies many items, but each item is supplied by only one supplier"

  → ***supplier*** *1:M* ***item***

- ***aouinati_store sells item:*** "the store sells many items, but each item is only sold in only one store which is the AOUINATI store"

  → ***aouinati_store*** *1:M* ***item***

N.B: You might find difficulties with understanding the 2 previous relationships, thinking that the relationship is many-to-many instead of one-to-many. However, keep in mind that the AOUINATI_STORE is one single entity and not many stores. We decided to create an entity called AOUINATI_STORE only because when interviewing the client, he was interested in posting store information on the internet. Therefore, creating and populating a store entity might be a great idea.

- ***customer is_registred in aouinati_store:*** "the store has information about many customers, but each customer is registered in only one store which is AOUINATI store"

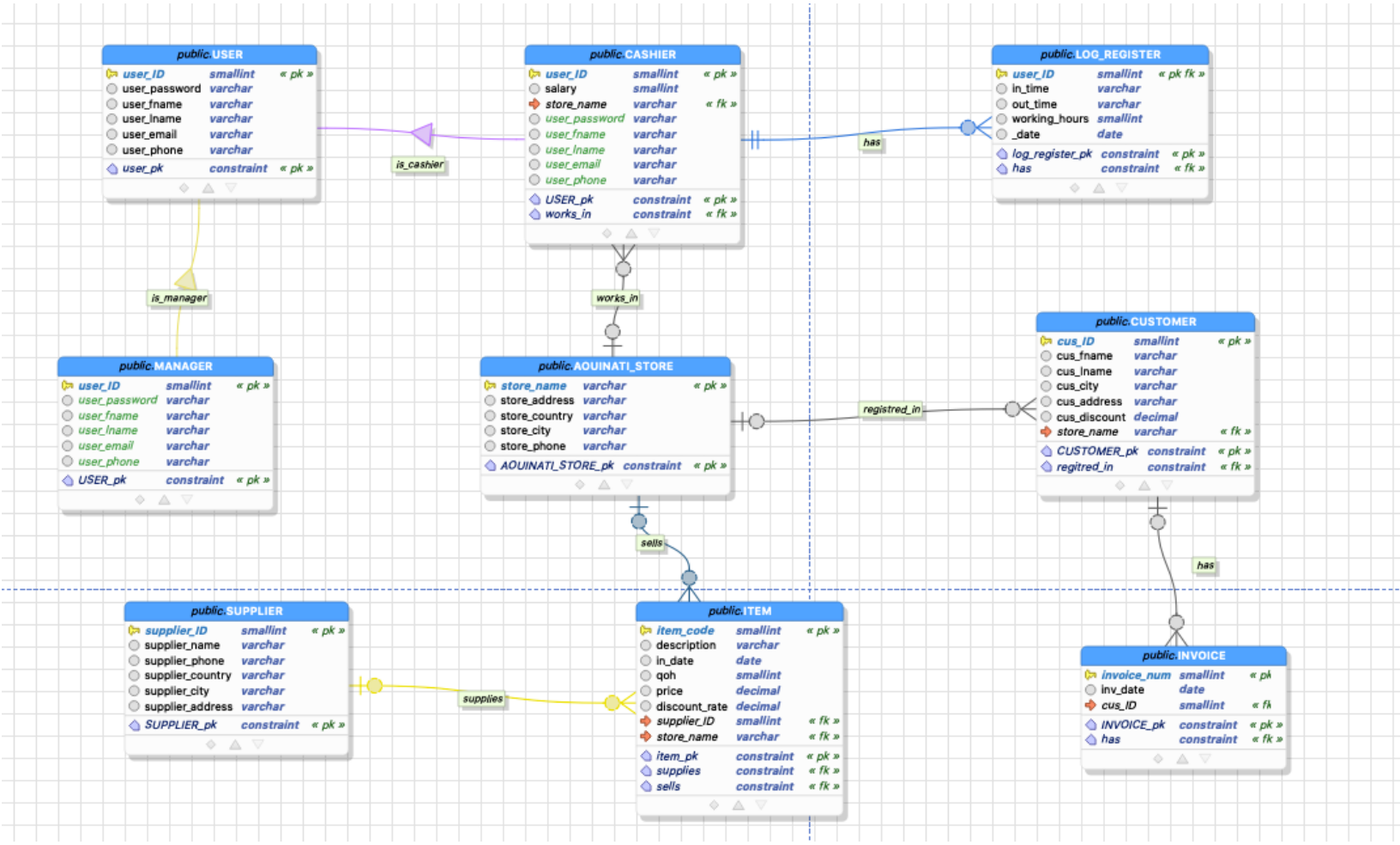
  → ***aounati_store*** *1:M* ***customer***

- ***customer has many invoices:*** "a customer could have many invoices while a single invoice is issued by only one customer"

  → ***cutomer 1:M invoice***

3) Initial Entity-Relationship Diagram:

According to the business rules and all of the previous above relationships, attributes and constraints, this is what we consider our ER diagram:



4) Table Normalization:

USER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is in BCNF** since determinants of the table ***(user_ID, user_email, user-phone),*** are all candidate keys.

→ **is in 4NF** since there are no ***multivalued dependencies.***

CASHIER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is in BCNF** since determinants of the table ***(user_ID, user_email, user-phone),*** are all candidate keys.

→ **is in 4NF** since there are no ***multivalued dependencies.***

MANAGER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is in BCNF** since determinants of the table ***(user_ID, user_email, user-phone),*** are all candidate keys.

→ **is in 4NF** since there are no ***multivalued dependencies.***

LOG_REGISTER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is already in BCNF** since there is only ***one candidate key, (user_ID).***

→ **is in 4NF** since there are no ***multivalued dependencies.***

CUSTOMER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is already in BCNF** since there is only ***one candidate key, (cus_ID).***

→ **is in 4NF** since there are no ***multivalued dependencies.***

INVOICE:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is already in BCNF** since there is only ***one candidate key, (invoce_num).***

→ **is in 4NF** since there are no ***multivalued dependencies.***

ITEM:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is already in BCNF** since there is only ***one candidate key, (item_code).***

→ **is in 4NF** since there are no ***multivalued dependencies.***

SUPPLIER:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is in BCNF** since determinants of the table ***(supplier_ID, supplier_phone),*** are all candidate keys.

→ **is in 4NF** since there are no ***multivalued dependencies.***

AOUINATI_STORE:

→ **is in 1NF** since there are no repeating groups, the primary key is identified, and all attributes are dependent on the primary key. There are no ***partial dependencies*** since the PK is not a composite key, and there are no ***transitive dependencies*** between the non-prime attributes of the table.

→ **is in 2NF** since there are no ***partial dependencies.***

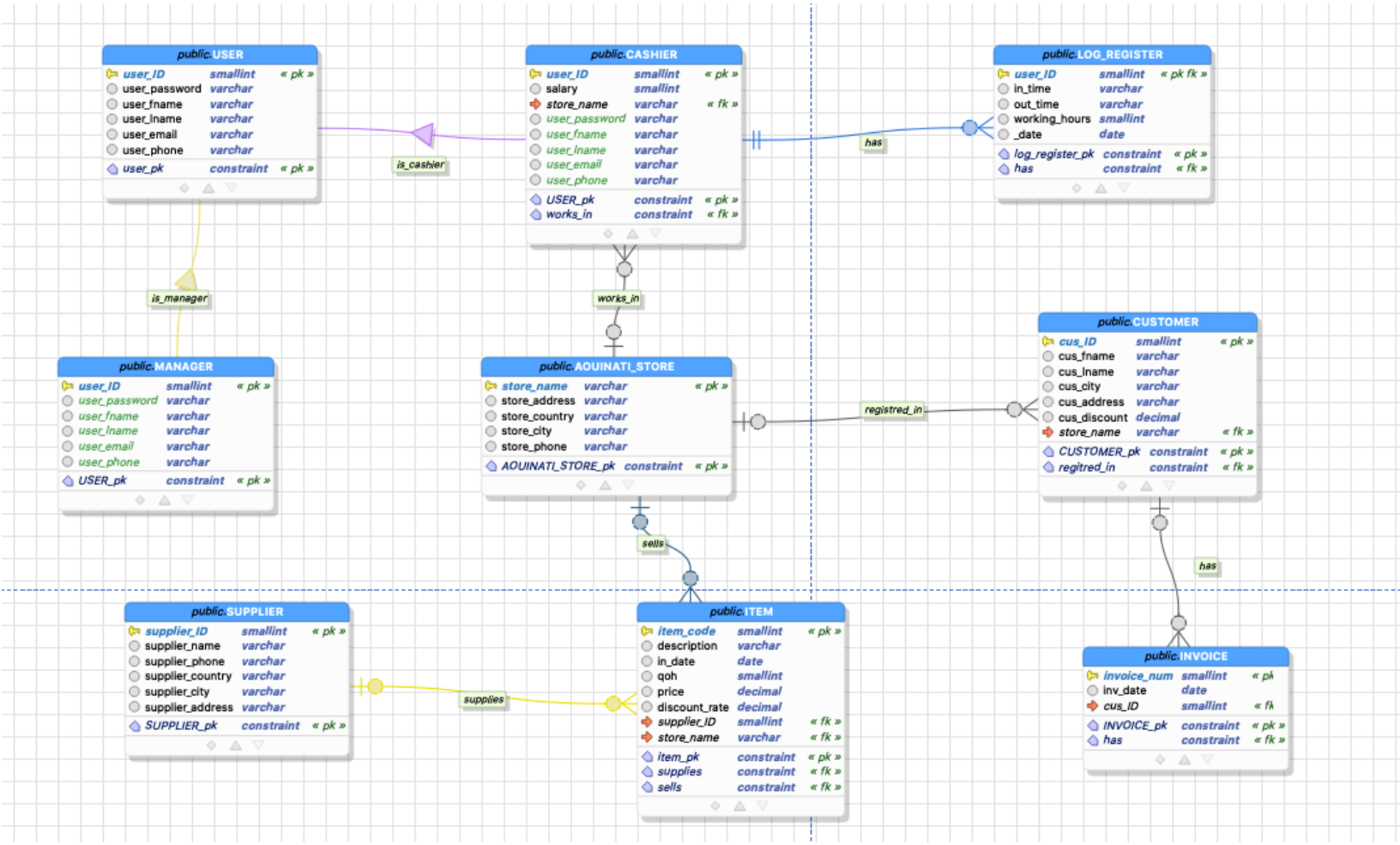
→ **is in 3NF** since there are no ***transitive dependencies.***

→ **is in BCNF** since determinants of the table ***(store_name, store_phone),*** are all candidate keys.

→ **is in 4NF** since there are no ***multivalued dependencies.***

5) Final Entity-Relationship model:

Concerning this part, we decided that our Entity-Relationship will be the same as the in-part c), the initial one, since we went through all steps of normalization, and we found that our ERD is already normalized. Therefore, our final Entity-Relationship model will be the following:



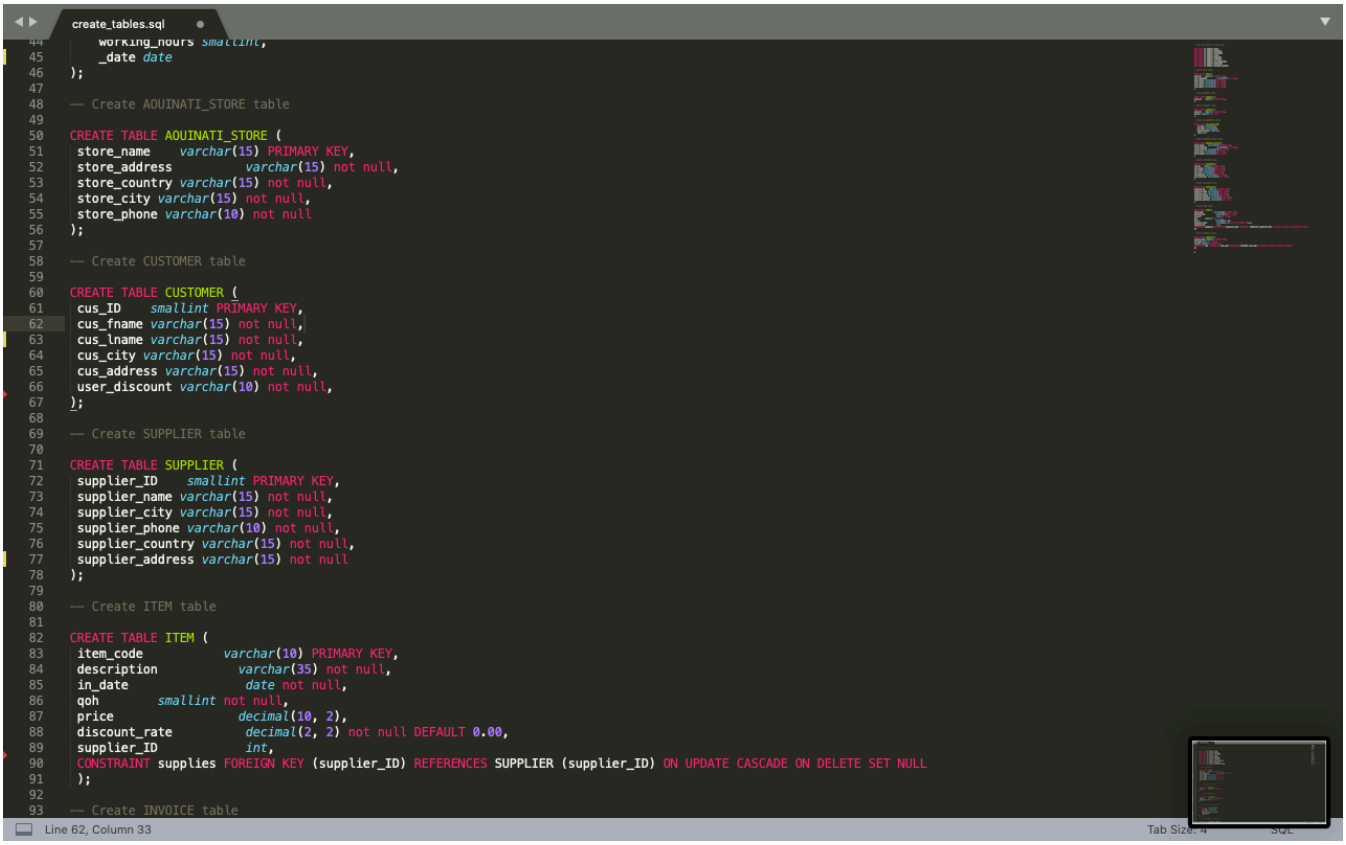
B) Logical design:

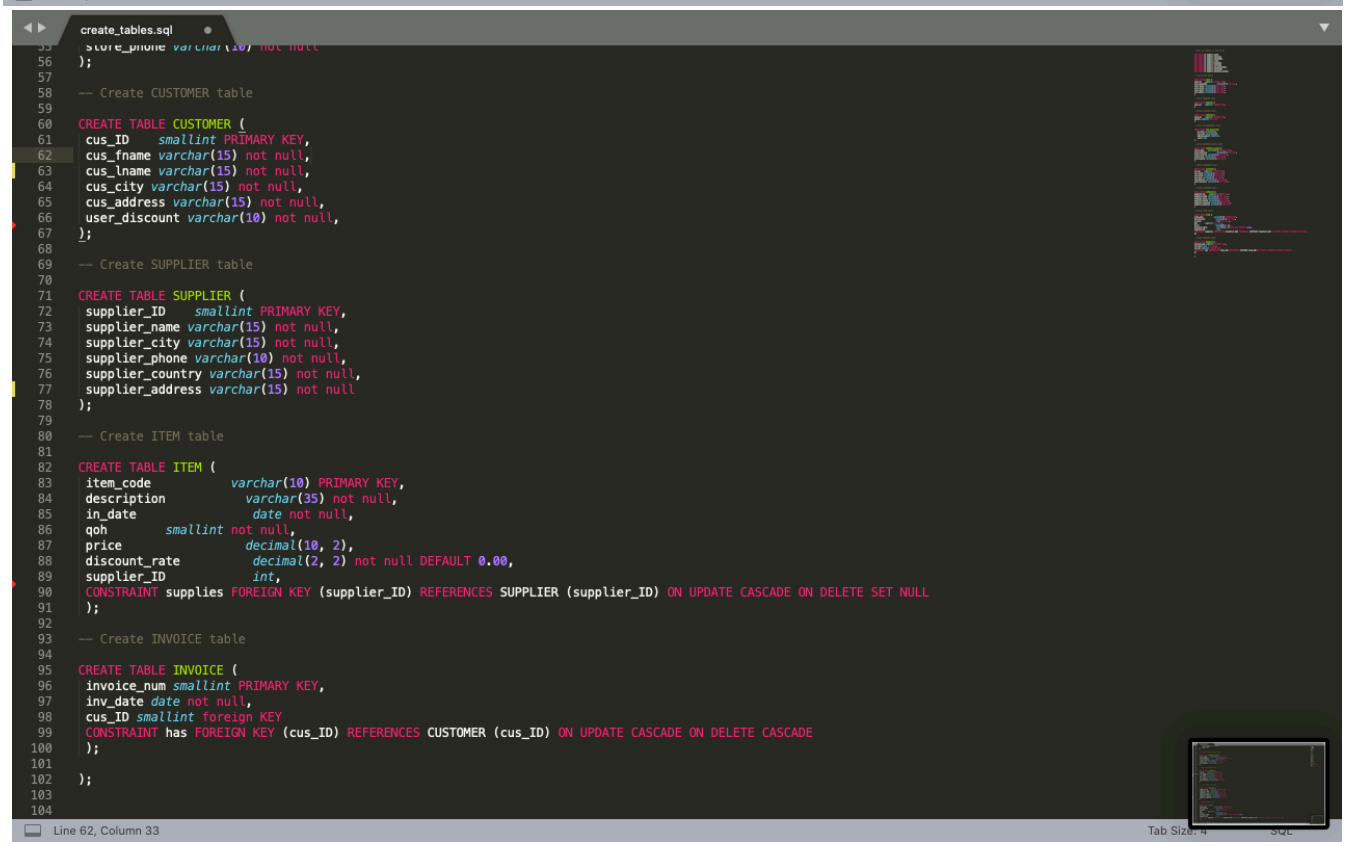
Concerning the implementation part, we first started by creating the tables using a text file where all the tables are created and populated, before moving on to pasting the code on pgadmin 4 software, in order to create and initialize our database. These are screenshots of the all the code needed:

create_tables.sql

```sql


-- Drop the tables if they exist

DROP TABLE IF EXISTS USER;
DROP TABLE IF EXISTS INVOICE;
DROP TABLE IF EXISTS CUSTOMER;
DROP TABLE IF EXISTS ITEM;
DROP TABLE IF EXISTS MANAGER;
DROP TABLE IF EXISTS CASHIER;
DROP TABLE IF EXISTS LOG_REGISTER;
DROP TABLE IF EXISTS SUPPLIER;
DROP TABLE IF EXISTS AOUINATI_STORE;

-- Create USER table

CREATE TABLE USER (
 user_ID    smallint PRIMARY KEY,
 user_password          varchar(15) not null,
 user_fname varchar(15) not null,
 user_lname varchar(15) not null,
 user_email varchar(15) not null,
 user_phone varchar(10) not null,
);

-- Create MANAGER table

CREATE TABLE MANAGER (
 user_ID    smallint PRIMARY KEY,
);

-- Create CASHIER table

CREATE TABLE CASHIER (
 user_ID    smallint PRIMARY KEY,
 salary smallint NOT null
);

-- Create LOG_REGISTER table

CREATE TABLE LOG_REGISTER(
    in_time varchar(10),
    out_time varchar(10),
    working_hours smallint,
    _date date
);

-- Create AOUINATI_STORE table

CREATE TABLE AOUINATI_STORE (
```

Line 62, Column 33 Tab Size: 4 SQL


create_tables.sql
working_hours smallint,
_date date
);
-- Create AOUINATI_STORE table
CREATE TABLE AOUINATI_STORE (
store_name varchar(15) PRIMARY KEY,
store_address varchar(15) not null,
store_country varchar(15) not null,
store_city varchar(15) not null,
store_phone varchar(10) not null
);
-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
cus_ID smallint PRIMARY KEY,
cus_fname varchar(15) not null,
cus_lname varchar(15) not null,
cus_city varchar(15) not null,
cus_address varchar(15) not null,
user_discount varchar(10) not null,
);
-- Create SUPPLIER table
CREATE TABLE SUPPLIER (
supplier_ID smallint PRIMARY KEY,
supplier_name varchar(15) not null,
supplier_city varchar(15) not null,
supplier_phone varchar(10) not null,
supplier_country varchar(15) not null,
supplier_address varchar(15) not null
);
-- Create ITEM table
CREATE TABLE ITEM (
item_code varchar(10) PRIMARY KEY,
description varchar(35) not null,
in_date date not null,
qoh smallint not null,
price decimal(10, 2),
discount_rate decimal(2, 2) not null DEFAULT 0.00,
supplier_ID int,
CONSTRAINT supplies FOREIGN KEY (supplier_ID) REFERENCES SUPPLIER (supplier_ID) ON UPDATE CASCADE ON DELETE SET NULL
);
-- Create INVOICE table
Line 62, Column 33
Tab Size: 4
SQL

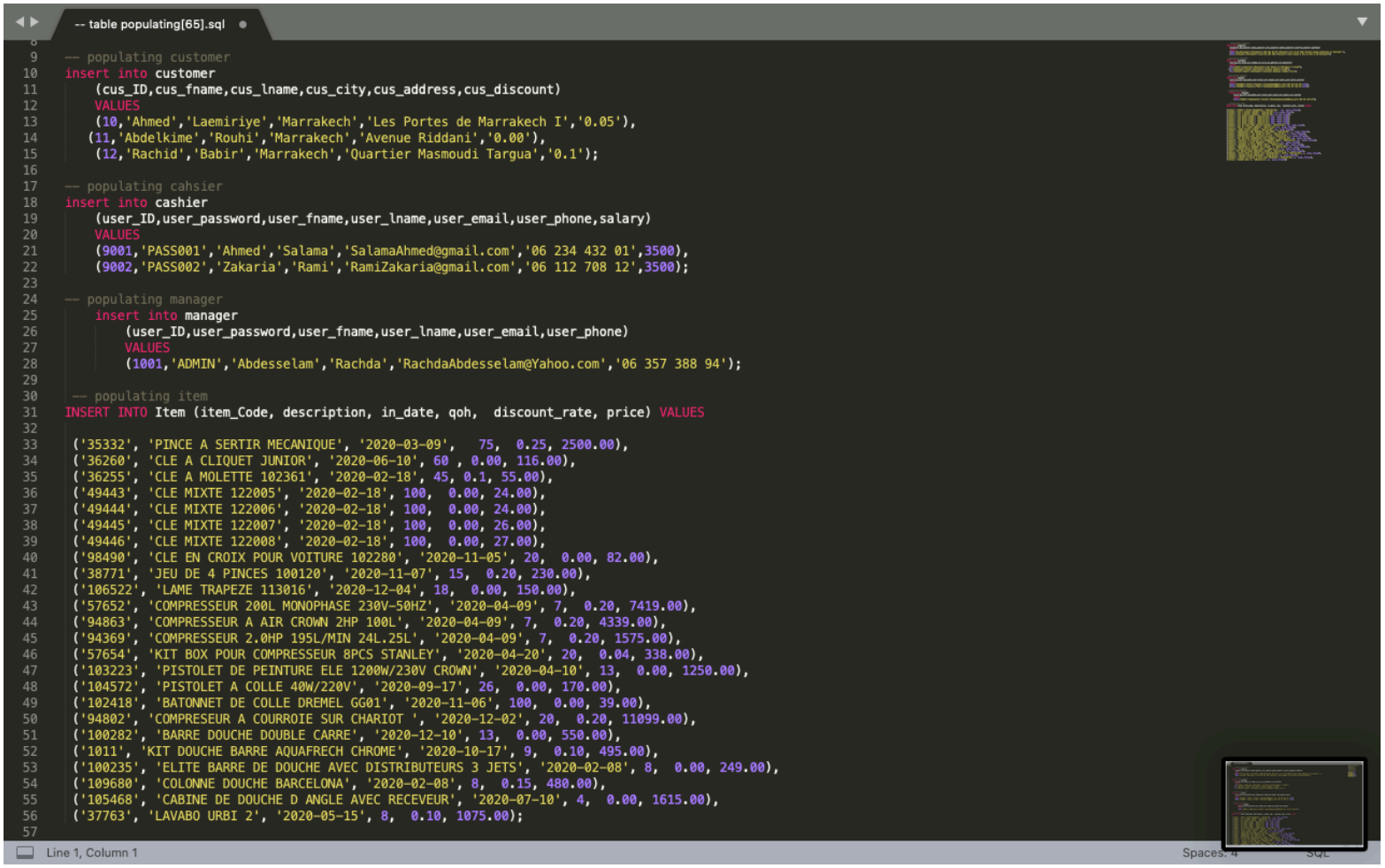

create_tables.sql
);
-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
cus_ID smallint PRIMARY KEY,
cus_fname varchar(15) not null,
cus_lname varchar(15) not null,
cus_city varchar(15) not null,
cus_address varchar(15) not null,
user_discount varchar(10) not null,
);
-- Create SUPPLIER table
CREATE TABLE SUPPLIER (
supplier_ID smallint PRIMARY KEY,
supplier_name varchar(15) not null,
supplier_city varchar(15) not null,
supplier_phone varchar(10) not null,
supplier_country varchar(15) not null,
supplier_address varchar(15) not null
);
-- Create ITEM table
CREATE TABLE ITEM (
item_code varchar(10) PRIMARY KEY,
description varchar(35) not null,
in_date date not null,
qoh smallint not null,
price decimal(10, 2),
discount_rate decimal(2, 2) not null DEFAULT 0.00,
supplier_ID int,
CONSTRAINT supplies FOREIGN KEY (supplier_ID) REFERENCES SUPPLIER (supplier_ID) ON UPDATE CASCADE ON DELETE SET NULL
);
-- Create INVOICE table
CREATE TABLE INVOICE (
invoice_num smallint PRIMARY KEY,
inv_date date not null,
cus_ID smallint foreign KEY
CONSTRAINT has FOREIGN KEY (cus_ID) REFERENCES CUSTOMER (cus_ID) ON UPDATE CASCADE ON DELETE CASCADE
);
);
Line 62, Column 33
Tab Size: 4
SQL

After that, we used the following code to populate and initialize our database, following the client's requests and data:

-- table populating[65].sql

```sql
--   populating supplier
insert into supplier
    (supplier_ID,supplier_name,supplier_city,supplier_phone,supplier_country,supplier_address)
    VALUES
    (0001,'Mr.Bricolage','Marrakech','080 203 80 80','Morocco','Lot 15-16 PARC Marjane Avenue Abdelkrim al Khattabi' ),
    (0002,'Bricoma','Marrakech','+212 524 336 700','Morocco','Sidi Ghanem 3 Rte de Safi N 16 Marrakech');

-- populating customer
insert into customer
    (cus_ID,cus_fname,cus_lname,cus_city,cus_address,cus_discount)
    VALUES
    (10,'Ahmed','Laemiriye','Marrakech','Les Portes de Marrakech I','0.05'),
   (11,'Abdelkime','Rouhi','Marrakech','Avenue Riddani','0.00'),
    (12,'Rachid','Babir','Marrakech','Quartier Masmoudi Targua','0.1');

-- populating cahsier
insert into cashier
    (user_ID,user_password,user_fname,user_lname,user_email,user_phone,salary)
    VALUES
    (9001,'PASS001','Ahmed','Salama','SalamaAhmed@gmail.com','06 234 432 01',3500),
    (9002,'PASS002','Zakaria','Rami','RamiZakaria@gmail.com','06 112 708 12',3500);

-- populating manager
    insert into manager
        (user_ID,user_password,user_fname,user_lname,user_email,user_phone)
        VALUES
        (1001,'ADMIN','Abdesselam','Rachda','RachdaAbdesselam@Yahoo.com','06 357 388 94');

 -- populating item
INSERT INTO Item (item_Code, description, in_date, qoh,  discount_rate, price) VALUES

 ('35332', 'PINCE A SERTIR MECANIQUE', '2020-03-09',   75,  0.25, 2500.00),
 ('36260', 'CLE A CLIQUET JUNIOR', '2020-06-10', 60 , 0.00, 116.00),
 ('36255', 'CLE A MOLETTE 102361', '2020-02-18', 45, 0.1, 55.00),
 ('49443', 'CLE MIXTE 122005', '2020-02-18', 100,  0.00, 24.00),
 ('49444', 'CLE MIXTE 122006', '2020-02-18', 100,  0.00, 24.00),
 ('49445', 'CLE MIXTE 122007', '2020-02-18', 100,  0.00, 26.00),
 ('49446', 'CLE MIXTE 122008', '2020-02-18', 100,  0.00, 27.00),
 ('98490', 'CLE EN CROIX POUR VOITURE 102280', '2020-11-05', 20,  0.00, 82.00),
 ('38771', 'JEU DE 4 PINCES 100120', '2020-11-07', 15,  0.20, 230.00),
 ('106522', 'LAME TRAPEZE 113016', '2020-12-04', 18,  0.00, 150.00),
 ('57652', 'COMPRESSEUR 200L MONOPHASE 230V-50HZ', '2020-04-09', 7,  0.20, 7419.00),
 ('94863', 'COMPRESSEUR A AIR CROWN 2HP 100L', '2020-04-09', 7,  0.20, 4339.00),
 ('94369', 'COMPRESSEUR 2.0HP 195L/MIN 24L.25L', '2020-04-09', 7,  0.20, 1575.00),
 ('57654', 'KIT BOX POUR COMPRESSEUR 8PCS STANLEY', '2020-04-20', 20,  0.04, 338.00),
 ('103223', 'PISTOLET DE PEINTURE ELE 1200W/230V CROWN', '2020-04-10', 13,  0.00, 1250.00),
 ('104572', 'PISTOLET A COLLE 40W/220V', '2020-09-17', 26,  0.00, 170.00),
 ('102418', 'BATONNET DE COLLE DREMEL GG01', '2020-11-06', 100,  0.00, 39.00),
 ('94892', 'COMPRESEUR A COURROIE SUR CHARIOT ', '2020-12-02', 20,  0.20, 11099.00)
```

Line 1, Column 1 Spaces: 4 SQL



```sql
-- table populating[65].sql

-- populating customer
insert into customer
    (cus_ID,cus_fname,cus_lname,cus_city,cus_address,cus_discount)
    VALUES
    (10,'Ahmed','Laemiriye','Marrakech','Les Portes de Marrakech I','0.05'),
   (11,'Abdelkime','Rouhi','Marrakech','Avenue Riddani','0.00'),
    (12,'Rachid','Babir','Marrakech','Quartier Masmoudi Targua','0.1');

-- populating cahsier
insert into cashier
    (user_ID,user_password,user_fname,user_lname,user_email,user_phone,salary)
    VALUES
    (9001,'PASS001','Ahmed','Salama','SalamaAhmed@gmail.com','06 234 432 01',3500),
    (9002,'PASS002','Zakaria','Rami','RamiZakaria@gmail.com','06 112 708 12',3500);

-- populating manager
    insert into manager
        (user_ID,user_password,user_fname,user_lname,user_email,user_phone)
        VALUES
        (1001,'ADMIN','Abdesselam','Rachda','RachdaAbdesselam@Yahoo.com','06 357 388 94');

 -- populating item
INSERT INTO Item (item_Code, description, in_date, qoh,  discount_rate, price) VALUES

('35332', 'PINCE A SERTIR MECANIQUE', '2020-03-09',   75,  0.25, 2500.00),
('36260', 'CLE A CLIQUET JUNIOR', '2020-06-10', 60 , 0.00, 116.00),
('36255', 'CLE A MOLETTE 102361', '2020-02-18', 45, 0.1, 55.00),
('49443', 'CLE MIXTE 122005', '2020-02-18', 100,  0.00, 24.00),
('49444', 'CLE MIXTE 122006', '2020-02-18', 100,  0.00, 24.00),
('49445', 'CLE MIXTE 122007', '2020-02-18', 100,  0.00, 26.00),
('49446', 'CLE MIXTE 122008', '2020-02-18', 100,  0.00, 27.00),
('98490', 'CLE EN CROIX POUR VOITURE 102280', '2020-11-05', 20,  0.00, 82.00),
('38771', 'JEU DE 4 PINCES 100120', '2020-11-07', 15,  0.20, 230.00),
('106522', 'LAME TRAPEZE 113016', '2020-12-04', 18,  0.00, 150.00),
('57652', 'COMPRESSEUR 200L MONOPHASE 230V-50HZ', '2020-04-09', 7,  0.20, 7419.00),
('94863', 'COMPRESSEUR A AIR CROWN 2HP 100L', '2020-04-09', 7,  0.20, 4339.00),
('94369', 'COMPRESSEUR 2.0HP 195L/MIN 24L.25L', '2020-04-09', 7,  0.20, 1575.00),
('57654', 'KIT BOX POUR COMPRESSEUR 8PCS STANLEY', '2020-04-20', 20,  0.04, 338.00),
('103223', 'PISTOLET DE PEINTURE ELE 1200W/230V CROWN', '2020-04-10', 13,  0.00, 1250.00),
('104572', 'PISTOLET A COLLE 40W/220V', '2020-09-17', 26,  0.00, 170.00),
('102418', 'BATONNET DE COLLE DREMEL GG01', '2020-11-06', 100,  0.00, 39.00),
('94802', 'COMPRESEUR A COURROIE SUR CHARIOT ', '2020-12-02', 20,  0.20, 11099.00),
('100282', 'BARRE DOUCHE DOUBLE CARRE', '2020-12-10', 13,  0.00, 550.00),
('1011', 'KIT DOUCHE BARRE AQUAFRECH CHROME', '2020-10-17', 9,  0.10, 495.00),
('100235', 'ELITE BARRE DE DOUCHE AVEC DISTRIBUTEURS 3 JETS', '2020-02-08', 8,  0.00, 249.00),
('109680', 'COLONNE DOUCHE BARCELONA', '2020-02-08', 8,  0.15, 480.00),
('105468', 'CABINE DE DOUCHE D ANGLE AVEC RECEVEUR', '2020-07-10', 4,  0.00, 1615.00),
('37763', 'LAVABO URBI 2', '2020-05-15', 8,  0.10, 1075.00);

```


Finally, we copied the previous codes and past in into the pgadmin 4 application, and our database was created, initialized and ready.

C) Physical design:

In the physical design part, we will consider and cover table storage space, access privileges and database roles.

Moreover, we believe that there should be only one role, which will the root user or the manager, that our team will help to create its own unique role, with all its important specifications. However, for the demo purpose, our team created a new role as well as a new database called “WeHelpYou_role” and “WeHelpYou_DB”.

Moving on to the storage specifications, again, since our database is small and only a demo, it can run on any computer hardware, even the customer could use this demo version on any existing platform. In fact, more about the storage details are discussed in the "Future work", the last part of the document.

Also, we believe that since the database is composed of only 9 tables and row are relatively few, there is no need for indexes.

VI. Implementation:

A) Database tables and population:

In this part, as we mentioned before, we copied and pasted it in the pgadmin4 IDE, in order to create the database, its tables. We also did the same concerning the population of the tables.

B) Application architecture:

When it comes to the application architecture of our project, we used the following tools and software we already dealt with in our CSC 3326:

- ***PgAdmin 4,*** for simpler database and tables creating and population.
- ***SublimeText,*** for better and simpler population code of our database.
- ***Netbeans,*** IDE for better java and web coe manipulation.
- ***JDK 15,*** used in netbeans IDE.
- ***JavaScript,*** for better and nicer web interfaces.

- ***Tomcat server,*** the server software we choose to operate on our database.

As indicated before, we used the sublime text software to write our database script, then we copied and pasted the file code into the pgadmin 4 interface, and we had our database created and initialized after also copying the population code.

We used Netbeans IDE, JDK 15 and TomCat Server after that to display the list of items, and the different functions we described and implemented in our web application.

VII. Testing and fine-tuning:

Finally, after describing all the steps, in this part we implemented and tested our web application. The following YouTube Video link demonstrate a short video:

https://youtu.be/O8dLQSVGQh4

You will find attached also the Netbeans project that you can run and manipulate all the functions existing on the website.

Also, you will find attached the creation and population of the tables code.

N.B: In Netbeans, right click on the main.html file as showed in the video, instead of running the whole project,

VIII. Conclusion:

Working on this project has enabled to us to grasp the class's material by applying it in the real world. Focusing both on, enhancing our soft skill when working with the client to create business rules and making sure that our design is the most suitable for all the users of this application. Far more, we enhanced our hard skill as well, and become versed in designing and implementing a database with multiple attributes and constraints, using SQL through JDBC while applying the basic functions, procedures, and triggers.

This experience has provoked us to learn about different languages such as CSS, HTLM, and JS with its multiple Fames to deliver the best product to our customer, which been an awestruck learning experience.

IX. Future Work:

We believe that our project is a first step towards the implementation of a largest database, in fact, our team put so much effort that we believe with some future work, it could really be implemented inside one or more shops / store. In fact, we believe that only some security

enhancements and slight JavaScript modifications are necessary, in order to fully implement a better working database.