AMERICAN UNIVERSITY OF BEIRUT

MAROUN SEMAAN FACULTY OF ENGINEERING & ARCHITECTURE

A DATABASE DESIGN FOR FOUR3THREE

By Group #2

Nourhan Salam (Group Leader)
(nas69@mail.aub.edu)
Aya Al Baba
(aaa270@mail.aub.edu)
Mohammad Salam
(mks46@mail.aub.edu)
Yehya Charif
(ymc09@mail.aub.edu)

A REPORT
submitted to Dr. Hussein Bakri in partial fulfillment of the requirements of phase 4 of the database project for the course EECE433 – Database Systems

November 2023

Table of Contents

1- Introduction

The owners of the company FOUR3THREE need a database for their online retail shop. This company would like to sell their clothing line as well as their cosmetic line. It's important to state that the online store's database should be cohesive, complete, and without any gaps to facilitate the work of the employees as well as the experience of the customers.

The aim is to create a database that will assist all parties that work in the company and interact with it. Starting off, the customers that would buy from this company are the ones that need wardrobe items or cosmetics, and all ages and genders are welcome to buy this company's products. This highlights the importance of having an efficient database which maximizes the satisfaction of all types of customers through all stages of shopping which include browsing the store, paying at the checkout, and tracking the order. The employees from different departments need this database to facilitate their work within and between departments, as well as operations with warehouses, where the items are produced and stored. The company needs to have different contractors to handle operations in their warehouses and different departments (e.g., producing the clothes, designing a work efficient space, keeping a clean and sanitized environment, commerce a marketing campaign).

These requirements should be satisfied by an ideal database for FOUR3THREE online retail store. This report provides an overview of the Entity-Relationship diagram used in designing the database for FOUR3THREE Company. The report starts with the references used to help design the ER diagram in section 2, followed by the tools used in section 3. In section 4, a detailed description of the requirements is provided. Section 5 contains a legend for all symbols used. In section 6, a complete view of the ER diagram is presented, and detailed explanations of every entity (section 6.1) and relationship (section 6.2) are put forth, and section 7 details all steps of the mapping process. Section 8 shows the tables, and section 9 shows table states with fictitious entries. All SQL queries used to create the tables, views, triggers, and insert data are present in section 10. Section 11 presents ten queries that solve certain problems. Finally, Section 12 shows the normalization process.

2- References

[1] R. A. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*. Boston i pozostałe: Pearson, 2017.

3- Tools Used to Draw the ER Diagram

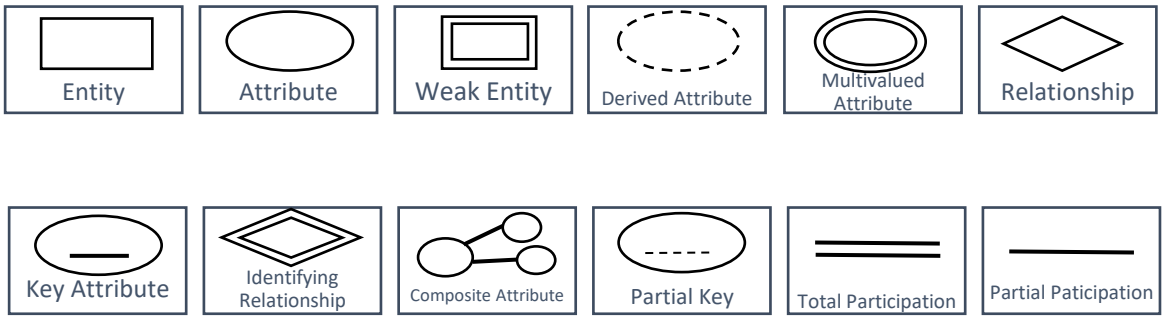
Draw.io provided the needed features which helped in creating a clear and readable ER diagram.

4- System Description and Requirements

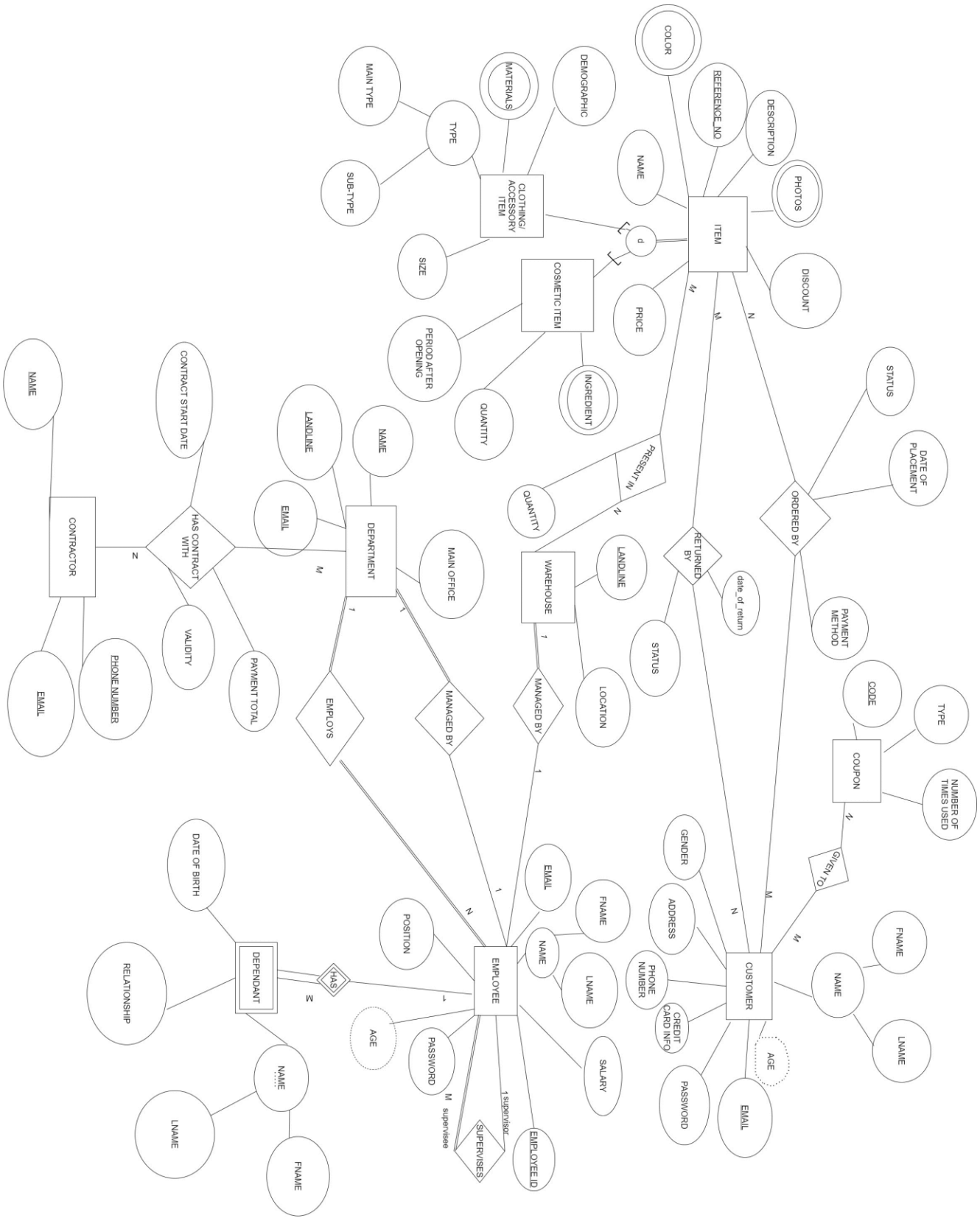
- An item can be of two types: clothing/ accessory items and cosmetics items. An item has a text description, name, price, a unique reference number, a discount percentage, as well as multiple colors and photos.
- A clothing/ accessory item is targeted towards a certain demographic, has a size, is composed of certain materials, and is classified according to a main type and a sub type (e.g.: main type: trousers and sub type: cargo pants)
- Cosmetics have several ingredients listed, in addition to the quantity of the product (e.g., 0.3 fluid ounces, 3 grams) and a PAO (period after opening) symbol.
- Every customer has a name composed of a first name and a last name. Their gender and date of birth are recorded. Upon signing up, their unique email is recorded, along with their password, phone number, credit card information, and address.
- A customer can order many items, and an item can be ordered by many customers. For every order, we note the status, payment method, and the date it was placed. Also, a customer can return an item within a 30-day period from the purchase date. A return instance has a status to know if it is exchanged, returned, or rejected.
- An employee has a first name and last name, a unique employee id and email, and a password that they choose. Furthermore, their position, salary, and date of birth are recorded.

- An employee can have many dependents. A dependent's name, date of birth, and their relationship to the employee are documented. An employee can be a supervisor to many employees, and each employee has a supervisor.
- As for a department, it has a unique name, email, and landline. It is located in a certain primary office.
- Each department is managed by one employee, and an employee may or may not be a manager. Every employee is employed in a certain department, and each department has several employees.
- The contractors must be stored. Contractors have a name and a unique phone number and email.
- Different departments can make contracts with contractors. For every contract, the contract start date, the contract's validity (length of the contract), and the payments that have been made should be documented.
- Warehouses have a location and a distinct landline. Items can be present in several warehouses, and the quantity in each warehouse must be specified. Every warehouse is managed by an employee, and an employee can manage only one warehouse.
- Each coupon has a distinct code and a certain type (e.g., % discount, free delivery, bundle offer). It can be given to several customers, and a customer can receive several coupons. The number of times a specific coupon has been used should be tracked.

5- Legend of ER Diagram Symbols

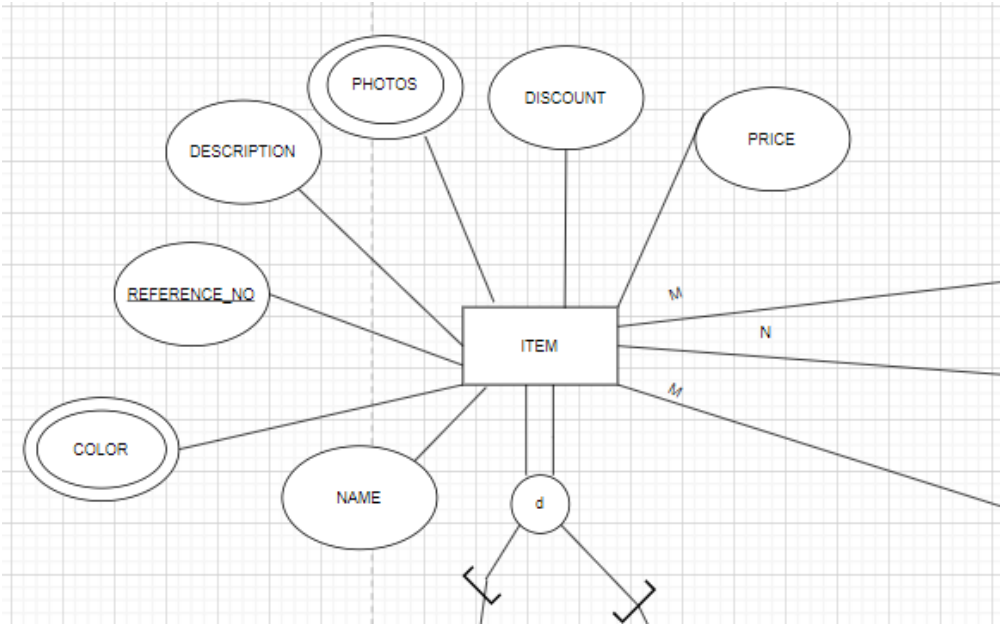


6- – New Complete Amended ER Diagram for FOUR3THREE:



6.1- Entity Types and Their Attributes

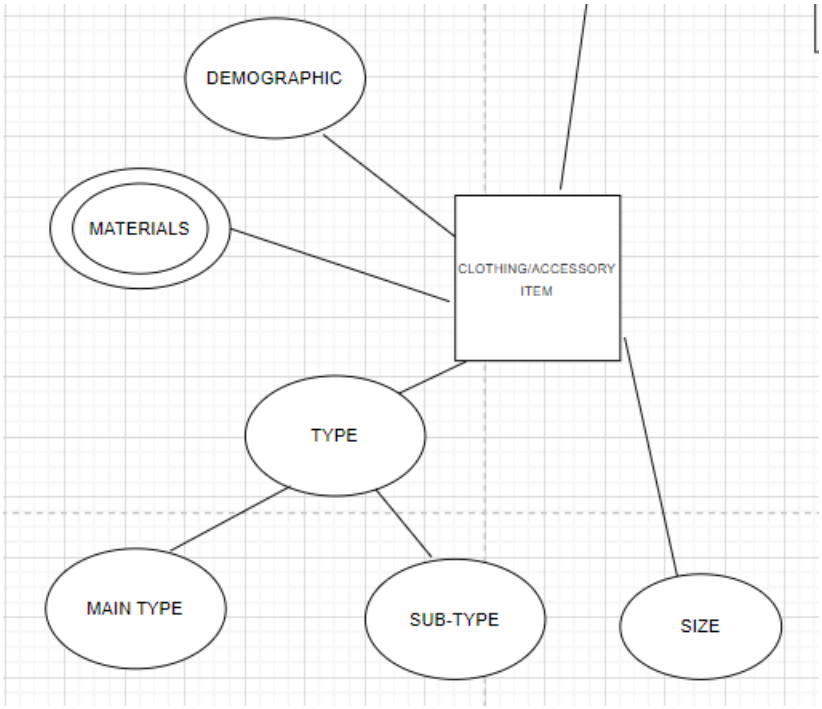
6.1.1- Item



The “Item” entity represents a product *type* that is produced and sold by this company. This product type is sold as many instances. The database must store every product to enable customers to browse through products on the online shop, and to help the company’s management to keep track of its items. Every item must be either a clothing/accessory item (see Section 6.1.2) or a cosmetics item (see section 6.1.3). Since these two entities have many common attributes and relationships, they are generalized into the “Item” entity to avoid redundancy in the database.

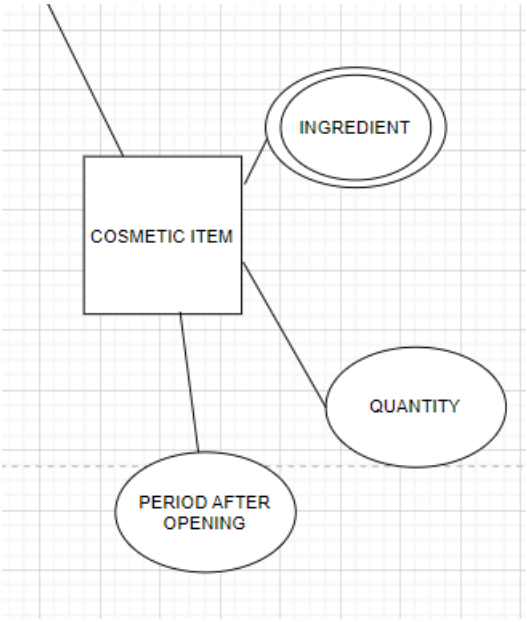
An item has a name so it can be easily identified and searched for by customers. Photos are necessary to showcase the product to customers, and they are present as a multivalued attribute because customers need to see the item from all its angles. The “color” attribute is multivalued because an item may be of different colors. “Discount” is present as an attribute because every item may have a different discount percentage as seen fit by the marketing department. The reference number is necessary for each product because it facilitates product identification in case two products have similar names, and it is selected as a key because it is unique for every product. The price of each item is stored because it is important for both customers and the management. A text description accompanies every item, and it should include care instructions and a description for marketing purposes.

6.1.2- Clothing/Accessory Item



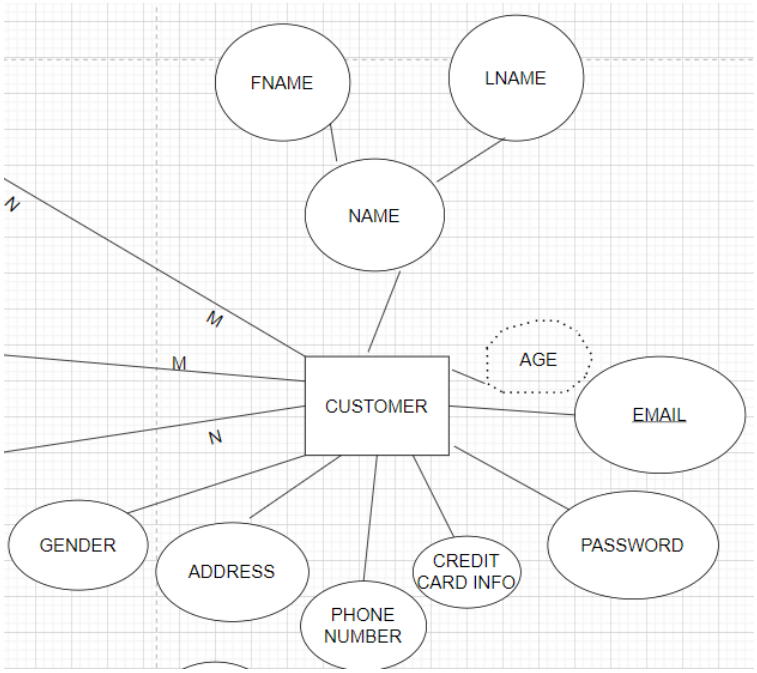
Clothes and accessories are grouped together in one entity because all of their attributes are common. This entity inherits all attributes and relationships of the “Item” entity. Each item of this type has a “type” attribute, which is composed of a “main type” and a “sub-type”. This is necessary for grouping items together, as it enables customers to apply search filters. For example, customers may filter their search to “shirts” (main type) if they want to view all shirts, and they may further filter their search to “blouses” (sub-type). This enhances a customer’s shopping experience. Each clothing/accessory item has a specific size and can be made of several materials (e.g., 70% cotton and 30% polyester). Moreover, it has a certain demographic (e.g., women, men, kids, unisex)

6.1.3- Cosmetics Item



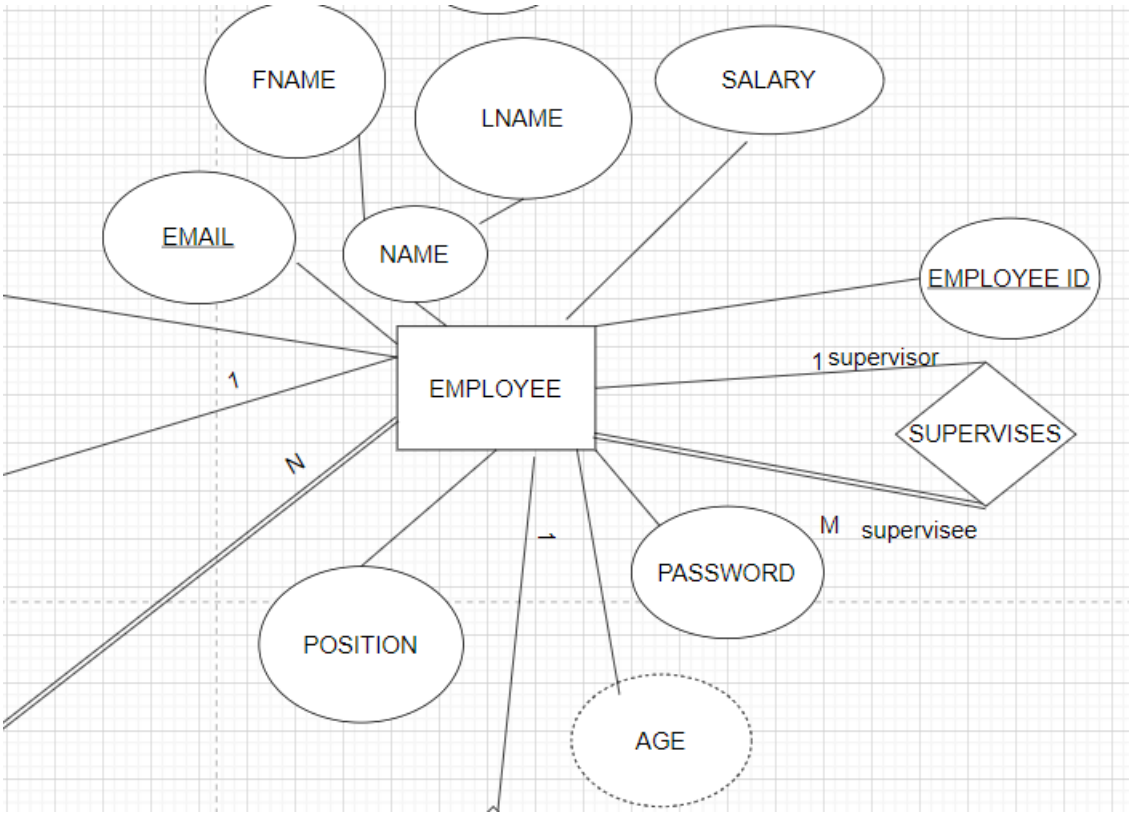
Cosmetics are items, so they inherit all attributes and relationships of the “Item” entity. The customers must be able to know the ingredients, quantity, and PAO (period after opening) of each makeup item. Each item of this type has a list of ingredients; hence, the “ingredients” attribute is multivalued. The quantity is the volume if the product is a liquid, while it is the mass if the product is a solid. The PAO symbol indicates how long the product lasts after being opened (e.g. 12M).

6.1.4- Customer



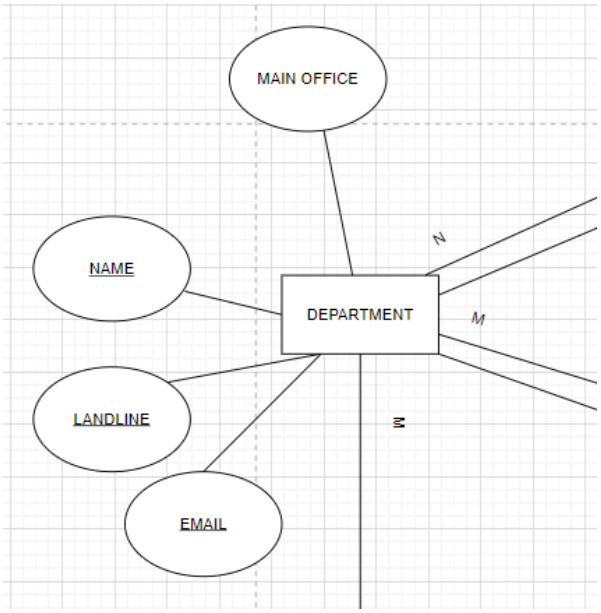
The company must keep track of customers to make better marketing decisions and analyze sales and customer behavior according to purchase history and demographics. This enables targeted promotions and marketing strategies that increase sales. Information related to demographics is in the form of gender and age, which is derived from “date of birth” to avoid frequently updating an “age” entry. Also, customers must have accounts to make orders and collect coupons based on their past purchases. An email is necessary to identify the customer, and it is used alongside the phone number by the customer service department to contact customers. Credit card information and an address are stored to facilitate purchases for customers, as they may not want to enter this information every time they make a purchase. A password is needed to log into an account. Every account has a name associated with it, and it is divided into “first name” and “last name”.

6.1.5- Employee



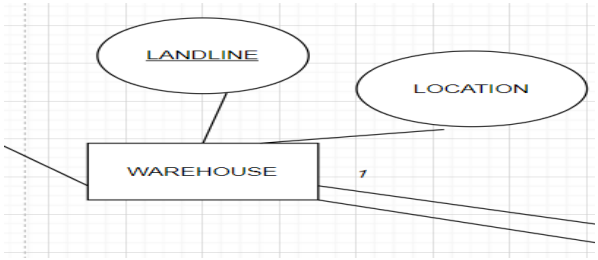
The company has employees who need to be tracked for management purposes. Each employee has a unique ID to distinguish employees from one another. Moreover, each employee has a unique email with a corresponding password which they would use to grant them more privileges when accessing the application than a normal customer. Finally, some information needs to be stored for each employee such as their date of birth, name which is divided into first name and last name, their position in the company, and the salary they earn for accounting purposes.

6.1.6- Department



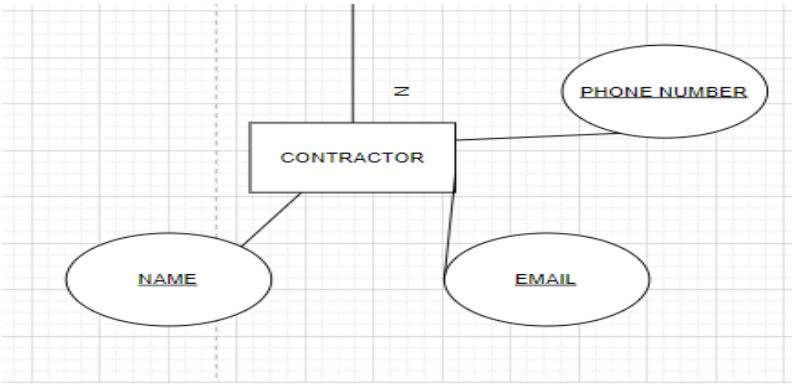
The company has different departments that handle different responsibilities to manage the company in a well-organized manner (e.g., customer service, accounting, marketing). Information about each department must be stored to facilitate the management's work. Each department has a unique name, a landline which is used for communication between employees or business calls, and an email which customers could use to report any problems or difficulties they are facing. A department also as a main office for employees to operate in.

6.1.7- Warehouse



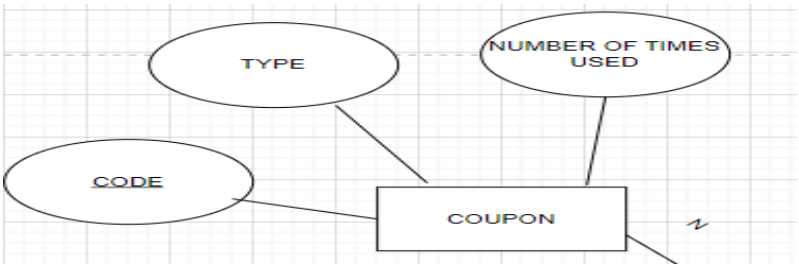
The company needs warehouses to store all its clothing, accessories, and cosmetics. Information about these warehouses must be stored to facilitate packaging orders for customers and making decisions concerning restocking. A warehouse's location is recorded as well as its unique landline which is used for communications between employees or for business calls.

6.1.8- Contractor



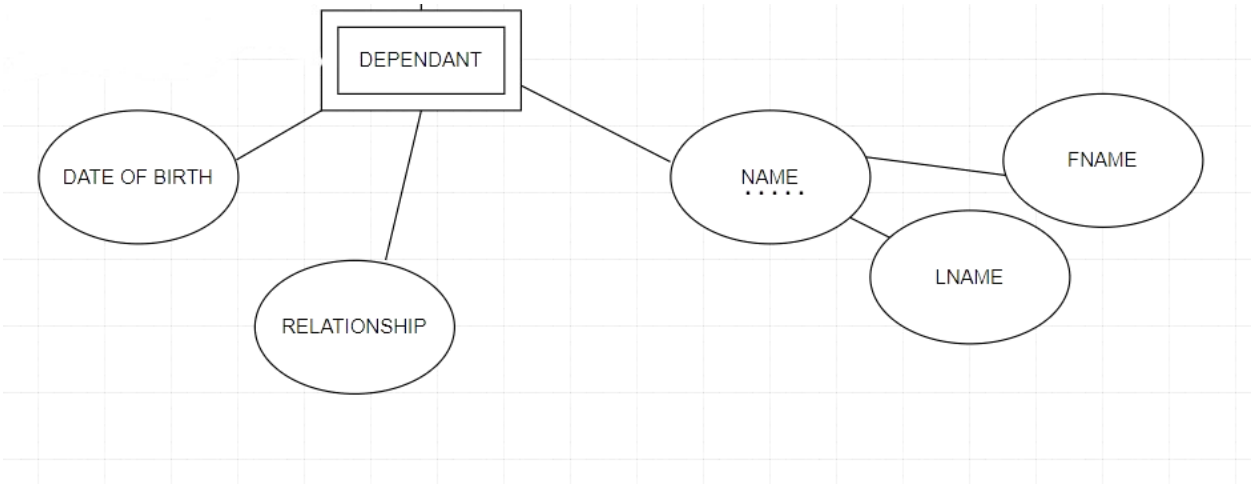
The company needs contractors to help in operations like managing the warehouses and performing the deliveries. These contractors must be stored in the database to keep track of them, as multiple contractors are needed for different operations. A contractor has a unique name, phone number, and email which helps the company stay in contact to help facilitate its needs and promptly assist the customers.

6.1.9- Coupon



Coupons must be stored to analyze sales and organize brand deals with influencers. The company provides coupons to customers which offer many types of benefits like a % discount, free shipping, offers, and bundles. This is saved as the “type” of a coupon. Each coupon has a unique code which is entered upon purchase. The number of times the coupon is used is tracked for customers to not pass a certain usage limit.

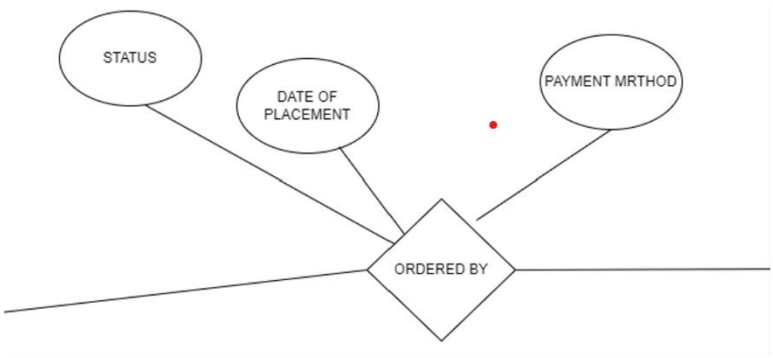
6.1.10- Dependent



The company offers benefits for employees with dependents to attract and retain employees, and these benefits vary depending on their relationship to the employee and their date of birth. Therefore, for each dependent, the company keeps track of their name, date of birth, as well as their relationship to the employee. This information should be readily available for accounting purposes.

6.2- Relationships and Their Explanations

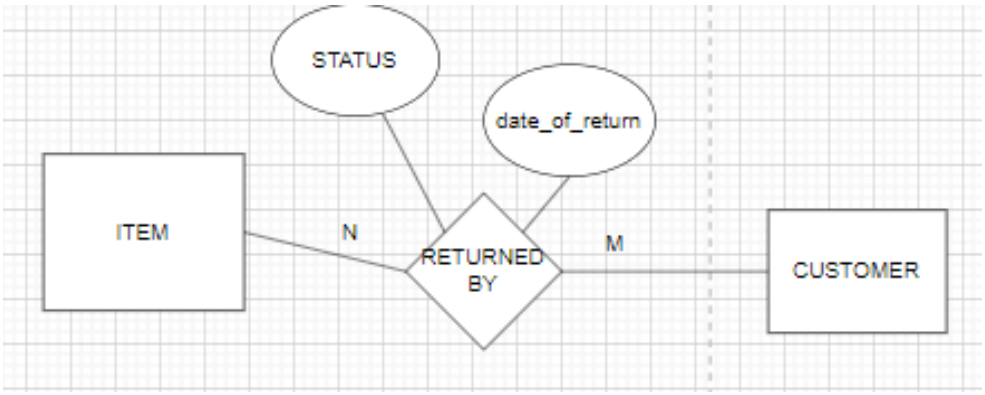
6.2.1- Ordered By



This relationship is present to keep track of orders by customers. An order has a status, which tells the customer the stage of the delivery (e.g., the order is being packed or shipped). Each order has a payment method (e.g., cash, card) to allow the party delivering it to know if a cash payment should be collected upon delivery. The date placed is noted to determine if an item can be returned at a given date, since returns are only allowed within a specified period.

As for the cardinality, customers may purchase an unlimited number of items, or they may have not made any purchases. An item can be sold to many customers (recall that “item” represents an item *type* which is sold as many instances).

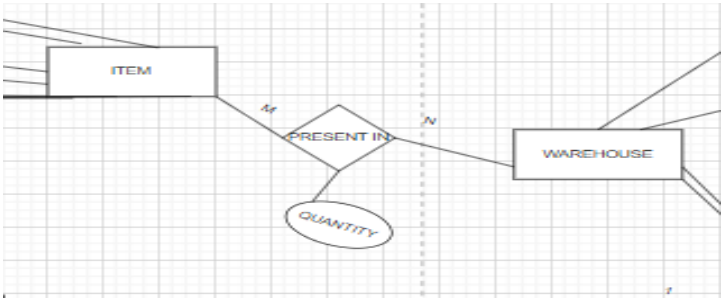
6.2.2- Returned By:



This relationship is required to keep track of items that customers returned. This information is needed by the management to tune targeted promotions and assess the performance of certain products based on how many of them are returned. Returns must be tracked because an employee must inspect the returned item to accept or reject the return request. Thus, a "status" attribute is needed to indicate that the request is pending, accepted, or rejected. Moreover, the date of the return request is needed to check if it is within 30 days or not.

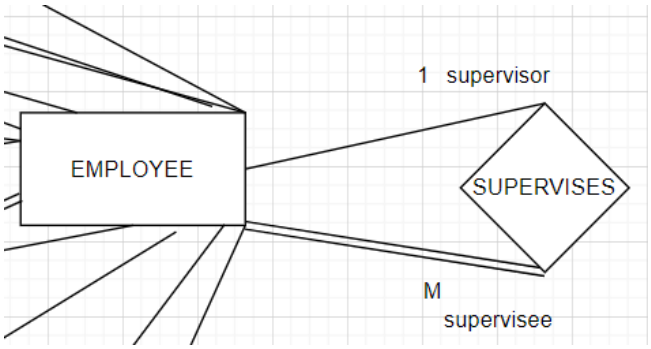
A customer may return an unlimited number of items, and an item may be returned by many customers.

6.2.3- Present In:



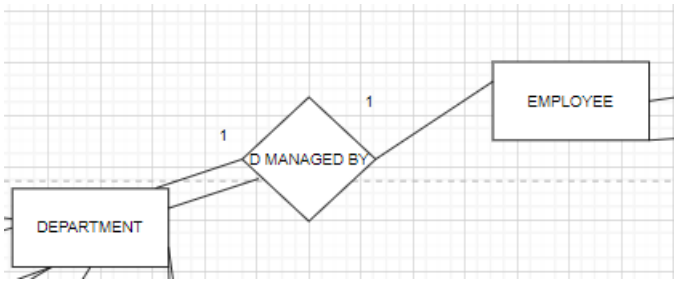
This relationship is needed to identify the warehouses an item is present in, in addition to the quantity present in a warehouse. This helps the management know if an item needs to be restocked and facilitates the packaging of orders. An item may be present in several warehouses, and a warehouse may contain many items.

6.2.4- Supervises:



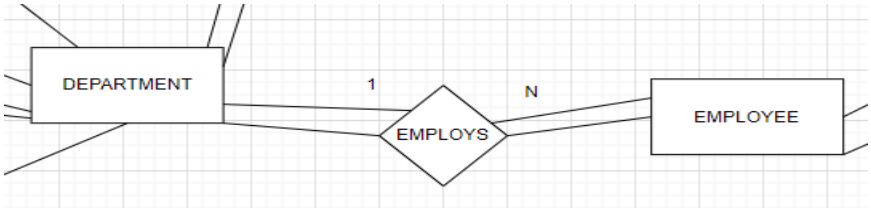
This relationship is for tracking the hierarchy of employees. An employee must have one supervisor, whereas a supervisor can supervise many supervisees. This helps the management identify the appropriate people to report to if any problem arises, and they would the supervisors of the concerned employees.

6.2.5- D Managed By:



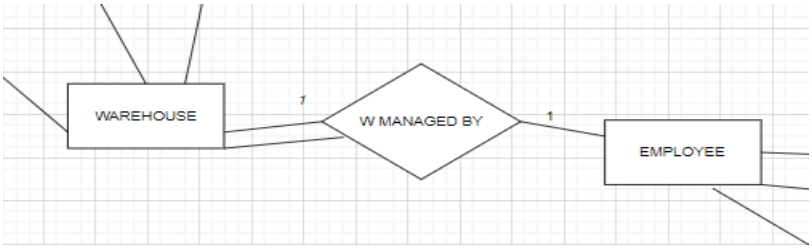
This relationship is to track which employee manages which department. As per the company's management policy, each department needs to be managed by one employee and an employee can only manage at most one department.

6.2.6- Employs:



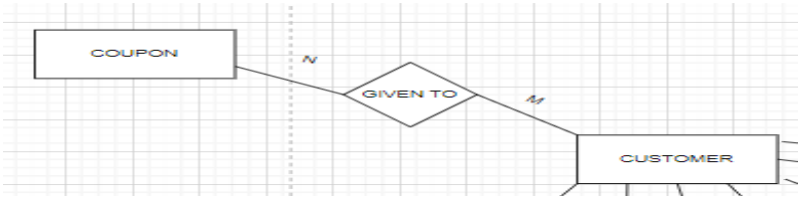
This relationship is to keep track of who works where, or in other words, which employee works in which department. All departments need employees; hence, each department employs many employees, and each employee works at a specific department.

6.2.7- W Managed By:



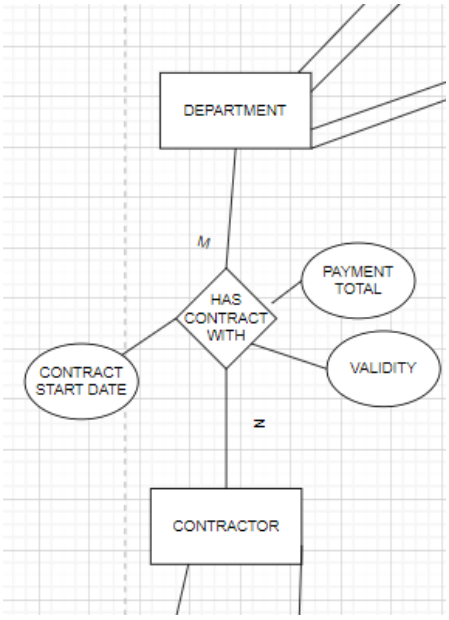
This relationship is to track which employee manages which warehouse. Such an employee must exist, as different departments may need to interact with the warehouse managers, and they should be able to identify who the manager of a warehouse is. Each warehouse needs to be managed by one employee and an employee can only manage at most one warehouse.

6.2.8- Given to:



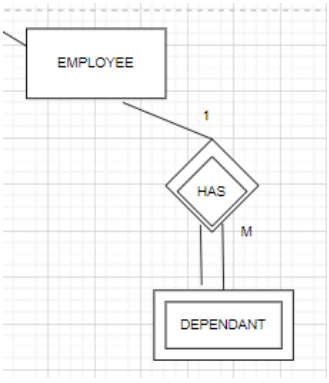
This relationship is to keep track of coupons given to customers to better decide when a coupon should be given to a customer, as they must not be distributed randomly. A coupon can be given to many different customers and each customer can receive many coupons.

6.2.9. Has a contract with:



This relationship is to keep a record of the contracts between different contractors and the departments for legal reasons. A contractor can have a contract with many different departments, and departments can have contracts with different contractors. The contract's start date, validity (how long the contract is still valid for), and total payment are recorded.

6.2.10. Has:



This relationship ties the employee to his/her dependents. As per company policy, an employee can have many dependents, while a dependent must be related to one employee.

7. Amended Description of ER to Relational Mapping

7.1. - Step 1 − Mapping of Strong Entity Types

In general, a relation must be created for every strong entity type, and it should include all atomic attributes of the corresponding entity type, and one of its keys is chosen as the primary key. This primary key cannot be null. However, the “Item” entity is an exception. It is a strong entity type that does not need a table because every item must be either a clothing/accessory item or a cosmetics item. Therefore, any item will be contained in the “Clothing/Accessory Item” or “Cosmetic Item” relations, but not in both simultaneously.

The “Clothing/Accessory Item” relation will have as columns all its atomic attributes, in addition to the attributes of the “Item” entity type, except for multivalued attributes. These attributes are: clothing_RefNb, clothing_name, clothing_price, clothing_discount, clothing_description, main_type, sub_type, size, and demographic. The reference number (clothing_RefNb) is the only key in this entity type, so it will be the primary key.

Similarly, the “Cosmetic Item” relation will have as columns its own atomic attributes in addition to those of the “Item” entity, except for multivalued attributes. Namely, these columns will be: cosmetic_RefNb, cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, and period_after_opening. The reference number (cosmetic_RefNb) will be the primary key, as it is the only key.

The “Customer” relation will have the following attributes: customer_email , FName, LName, password, gender, address, phone_nb, credit_card_info, and date_of_birth, which we derive age from. The only key for this entity type is “customer_email”, so it will be chosen as the primary key.

As for the “Employee” relation, its attributes will be: employee_ID, employee_email, salary, Fname, Lname, password, date_of_birth, and position. This entity type has two keys (employee_ID and employee_email). “Employee_ID” is arbitrarily chosen to be the primary key.

The “Department” relation’s columns will be: department_name, department_landline, email, and main_office. Of its three keys (department_name, department_landline, email), department_name is arbitrarily chosen as its primary key.

The “Warehouse” relation will have the following columns: landline, and location. Its only key is “landline”, so it will be the primary key.

The “Contractor” relation’s attributes are contractor_name, email, and contractor_phone_number. All of its attributes are keys, and the “contractor_name” is arbitrarily selected as the primary key.

The “Coupon” relation will have the following columns: code, type, and nb_of_times_used. The code is the only key, so it will be the primary key.

7.2- Step 2 – Mapping of Weak Entity Types

A relation must be created for a weak entity type, and it should include all its simple attributes. Moreover, it should contain a foreign key that references the primary key of the relation that represents the owner of this weak entity. In this case, a relation is created for the “Dependent” weak entity type. Its attributes that must be included are date_of_birth, Fname, Lname, and relationship. The owner entity is “Employee”, and its relation’s primary key is “employee_ID”. This is included as a foreign key in the “Dependent” relation. The primary key of this relation will be composed of “employee_ID”, and the partial key “name”, which is composed of “Fname” and “Lname”. All atomic attributes within the primary key cannot be null.

7.3 – Step 3 – Mapping of Binary 1:1 Relationship Types

For binary 1:1 relationship types, there are three approaches for mapping. We are going to use the foreign key approach: we will include a foreign key in one relation, and this foreign key will reference the primary key of the other relation. When applicable, the foreign key will be in the table of the entity that totally participates in the relationship.

The “W managed by” relationship between employee and warehouse is a 1:1 relationship. Therefore, we will add a foreign key to warehouse named “manager_ID” which references the primary key of the employee relation.

The “D managed by” relationship exists between employee and department which is also a 1:1 relationship. We will add a foreign key to department named “manager_ID” which references the primary key in the employee relation.

7.4 – Step 4 – Mapping of Binary 1:N Relationship Types

For each binary 1:N relationship, let R be the relation of the entity type which is at the N-side of the relationship, and S be that of the entity at the 1-side. We include in R a foreign key which references the primary key of S.

The “Supervises” relationship is between “employee” and itself, hence we will add a foreign key to “employee” named “supervisor_ID”.

The “Employs” relationship is between “employee” and “department”. The participating entity on the N-side in this relationship is “employee”; hence, we will add the primary key of “department” as a foreign key to “employee”, and this foreign key will be named “department_name”.

The “has” relationship is between “employee” and “dependent”. The participating entity on the N-side in this relationship is “dependent”; hence, we will add the primary key of “employee” as a foreign key to “dependent”, and it will be named “employee_ID”.

7.5 – Step 5 – Mapping of Binary M:N Relationship Types

Each binary many to many relationship needs a table to connect the participating entities, and it should include the primary key of each participating entity as a foreign key. The combination of these foreign keys will represent the primary key of the table. Hence, each table will consist of the foreign keys referencing the tables of the participating entities as well as the simple attributes of the many to many relationship type.

The "ordered by" relationship between customer and item will be mapped to two tables: one for "Clothing/Accessory Item" and another for "Cosmetic Item". The former table will be named "Ordered By-Clothing" , and it will contain the columns: clothing_RefNb, customer_email, status, date_of_placement, and payment_method. The latter table will be named "Ordered By-Cosmetic", and it will contain the columns: cosmetic_RefNb, customer_email, status, date_of_placement, and payment_method. The combination of clothing_RefNb, customer_email, and date_of_placement will make up the primary key for the first table, and the combination of cosmetic_RefNb, customer_email, and date_of_placement will make up the primary key for the second table. Note that in these two tables, cosmetic_RefNb references the primary key of "Cosmetic item", "clothing_RefNb" references that of "Clothing/Accessory item", and "customer_email" references that of "Customer".

The "returned by" relationship between customer and item will also have two tables: one for "Clothing/Accessory Item" and another for "Cosmetic Item". The first table will be named "Returned By-Clothing", and it will contain the columns: clothing_RefNb, customer_email, date_of_return, and status. The latter table will be named "Returned By-Cosmetic", and it will also contain the same columns, with cosmetic_RefNb instead of clothing_RefNb. The combination of clothing_RefNb, customer_email, and date_of_return will make up the primary key in the first table, and the combination of cosmetic_RefNb, customer_email, and date_of_return will make up the primary key for the second table. Also note that in these two tables, cosmetic_RefNb references the primary key of "Cosmetic item", "clothing_RefNb" references that of "Clothing/Accessory item", and "customer_email" references that of "Customer".

Similarly, the "present in" relationship between warehouse and item will be split into two tables: one for the "Clothing/Accessory Item" entity, and the other for the "Cosmetic Item" entity. The first table will be named " Present In-Clothing", and it will contain the columns: clothing_RefNb, landline, and quantity. The latter table will be named "Present In-Cosmetic", and it will contain the columns: cosmetic_RefNb, landline, and quantity. The combination of clothing_RefNb and landline will make up the primary key in the first table, and the combination of cosmetic_RefNb and landline will make up the primary key for the second table. Note that in each of these two tables, cosmetic_RefNb references the primary key of "Cosmetic item", "clothing_RefNb" references that of "Clothing/Accessory item", and "landline" references that of "Warehouse".

The “given to” relationship between customer and coupon will be mapped to a table that contains these columns: “customer_email” (references the primary key of “Customer”) and ”code” (references the primary key of “Coupon”). The combination of these columns will make up its primary key.

The “has contract with” relationship between department and contractor will be mapped to a table that contains the following columns: “dept_name”(references the primary key of “Department”), ”cont_name”(references the primary key of “Contractor”), ”contract start date”, ”validity”, and ”total payment”. The combination of “dept_name” and “cont_name” will be its primary key.

7.6 – Step 6 – Mapping of multivalued attributes

For each multivalued attribute, a relation should be created where its columns consist of the attribute and the primary key of the entity it belongs to (as a foreign key), and the combination of these columns will be the primary key of this relation.

The multivalued attribute “photos” which belongs to the item entity will have two tables: one for “Clothing/Accessory Item” and one for “Cosmetic Item”. The first will be called “photos-clothing” and will have these columns: “photo” and ”clothing_RefNb”, and both of these columns will form its primary key. “clothing_RefNb” is a foreign key, referencing “Clothing/Accessory item”. The second will be called “photos-cosmetic” and will have these columns: “photo” and “cosmetic_RefNb”, and both these columns will form its primary key. “cosmetic_RefNb” is a foreign key that references “Cosmetic item”.

The multivalued attribute “color” which belongs to the item entity will have two tables: one for “Clothing/Accessory Item” and one for “Cosmetic Item”. The first will be called “color-clothing” and will have the columns “color” and ”clothing_RefNb” (clothing_RefNb references “Clothing/Accessory item”), and both these columns will form its primary key. The second will be called “photos-cosmetic” and will have the columns “color” and “cosmetic_RefNb” (cosmetic_RefNb references “Cosmetic item”), and both these columns will form its primary key.

The multivalued attribute “materials” which belongs to the clothing/accessory item entity will have a table called “color” having the columns “color” and “clothing_RefNb” (clothing_RefNb references “Clothing/Accessory item”), and the combination of both columns will be its primary key.

The multivalued attribute “ingredients” which belongs to the cosmetic item entity will have a table called “ingredients” having the columns “ingredient” and “cosmetic_RefNb”(cosmetic_RefNb references “Cosmetic item”), and the combination of both columns will be its primary key.

7.7 – Step 7 – Mapping of N-ary relationships

For N-ary relationships, a relation should be created containing the primary keys of all entities participating in the relationship as foreign keys, in addition to all the attributes associated with the relationship. In the FOUR3THREE ER diagram, there are no N-ary relationships.

7.8 – Step 8 Mapping aggregation, specialization/generalization relationships

For relationships involving the item entity, which is a generalization of clothing/accessory items and cosmetic items, two tables must be created: one for clothing/ accessory items and another for cosmetic items. All these relationships in this ER are many to many relationships, and their mapping is detailed in section 8.5. Namely, they are "ordered by", "returned by", and "present in". The "Item" entity will not have a table, as explained in section 8.1. Instead, all the attributes in items (discount, price, name, reference number, and description) will be included in the tables of "clothing/accessory item" and "cosmetic item". Section 8.1, which discusses all strong entities, contains the details about mapping the "clothing/accessory item" and "cosmetic item" entities.

8. Final Display – All Tables

We show the primary keys in the following tables as underlined and the foreign keys as italic:

Clothing/Accessory item

(clothing_RefNb, clothing_name, clothing_price, clothing_discount, clothing_description, main_type, sub_type, size, demographic)

Cosmetic item

(cosmetic_RefNb, cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

Ordered By Clothing

(*clothing_RefNb, customer_email* date_of_placement, status,paytment_method)

Ordered By Cosmetic

(*cosmetic_RefNb, customer_email,* date_of_placement, status, payment_method)

Returned By clothing

(*clothing_RefNb, customer_email,*date_of_return,status)

Returned By cosmetic:

(*cosmetic_RefNb, customer_email,* date_of_return, status)

Present In Clothing

(*clothing_RefNb, landline,* quantity)

Present In Cosmeticx

(*cosmetic_RefNb, landline,* quantity)

Customer

(customer_email , FName, LName, password, gender, address, phone_nb, credit_card_info, date_of_birth)

Coupon Given To

(*customer_email, code*)

Coupon

(code, type, nb_of_times_used)

Employee

(employee_ID, employee_email, salary, Fname, Lname, password, date_of_birth, position, *supervisor_ID, department_name*)

Dependent

(*employee_ID,* Fname, Lname, date_of_birth, relationship)

Warehouse

(landline, *manager_ID,*location)

Department

(department_name, *manager_ID,* department_landline,email, main_office)

Has Contract With

(*department_name, contractor_name,* contract_start_date, validity, total payment)

Contractor

(contractor_name, email, contractor_phone_number)

Photos Cosmetic

(*cosmetic_RefNb* ,cosmetic_ photos)

Photos Clothing

(*clothing_RefNb,* clothing_photos*)*

Color Cosmetic

(*cosmetic_RefNb* ,cosmetic_ color)

Color Clothing

(*clothing_RefNb,* clothing_color*)*

Material

(*clothing_RefNb,* material)

Ingredients

(*cosmetic_RefNb,* ingredients)

9. Table States

Employee:

Employee ID	Employee email	salary	Fname	Lname	password	Date of birth	position	Supervisor ID	Dept name
200123	Tk19@433.com	5000	Tanaka	Miyazono	4g^&3e	07/01/1990	IT Officer	200128	IT
200124	Ma66@433.com	8000	Mohamad	Abbas	^&gsaur	09/02/1960	Business Analyst	200126	Finance

200125	Ab26@433.com	6000	Ali	Baba	^&geja	31/06/1988	Senior Fashion Designer	200130	Design
200126	Pm55@433.com	6000	Piers	Morgan	(*Y&dk	21/02/1998	Senior Business Analyst	200130	Finance
200127	Ae87@433.com	5000	Abbas	Al Nouri	JWGgrw	02/08/2000	Customer Service Manager	200129	Customer service
200128	Jw66@433.com	7000	Jacinda	White	JAauwgr	04/10/1977	IT Manager	200129	IT
200129	As32@433.com	20000	Alice	Shields	OAUsew!!	02/04/1980	CEO	200130	General Manage-ment
200130	Jj89@433.com	20000	Janice	Jackson	7!!u=*&	01/01/1966	Sales Executive	200129	Sales
200131	Jb00@433.com	9000	Jelena	Brown	Yewug&%	08/01/1989	Marketing Officer	200129	Marketing

Dependent:

employee_ID	Fname	Lname	Date_of_birth	relationship
200124	Emily	White	03/12/2009	daughter
200128	William	Blanchard	21/09/1935	father
200126	Ada	Shields	01/01/1960	mother
200126	Samer	El Masri	20/01/2023	son
200131	Alice	Brown	10/10/2020	daughter
200125	Ali	Jackson	01/03/2010	son
200123	Kaori	Miyazono	04/04/2022	daughter
200129	Tim	Joyce	31/08/2020	son
200126	Karen	Spencer	03/03/1998	wife
200125	Susan	Macey	01/05/1990	wife

Warehouse:

landline	Manager_ID	Location
01345632	200123	Ain El Remmaneh

01683444	200124	Dekwaneh
09527997	200127	Byblos
05188927	200126	Byblos
05836986	200128	Baabda
009661234778	200129	Riyadh
009661938224	200131	Jeddah
09352876	200125	Kesserwan

Department:

Department name	Manager ID	Department landline	email	Main office
IT	200128	01357261	it@433.com	Sin El Fil
Finance	200126	01746245	finance@433.com	Sin El Fil
Design	200125	01726252	design@433.com	Clemenceau st.
Customer service	200127	01888222	customerservice@433.com	Mathaf
General Management	200129	01987567	management@433.com	Northern Metn
Sales	200130	01720000	sales@433.com	Beirut Central District
Marketing	200131	09123123	marketing@433.com	Metn

Contractor

Contractor name	email	Contractor phone number
GASE specialties	gasesp@gase.com	05052533
Sbeity contracting	Sbeity@sbeity.com	09473622
Compass contracting	compass@comp.com	08976252
ENERGYTIKA SARL	energytika@energ.com	01999676

Yamen Est.	yamen@yamen.com	01878888
Beet General Contractors	beet@bgc.com	01828999
FYCO SARL	fyco@fyco.com	09888777
SMART Contactors	smartcontractors@smart.com	01468999
TRUST Contracting	trustcontracting@trust.com	09444555
BFD General Contractors	bfdcontracting@bfd.com	009661897226

Has contract with:

Department name	Contractor name	Contract start date	validity	Total payment
Finance	Compass Contracting	02/01/2015	10 years	30000
Design	FYCO SARL	03/03/2020	4 years	10000
Finance	Yamen Est	09/08/2021	3 years	10000
Sales	Beet General Contractors	02/01/2023	2 years	4000
IT	Sbeity contracting	05/05/2019	6 years	20000
General Management	TRUST Contracting	01/01/2023	2 years	3000
Design	GASE specialties	12/12/2018	5 years	9000
General Management	Compass Contracting	31/08/2022	2 years	8500
Finance	SMART Contractors	15/01/2018	5 years	9500
Design	Yamen Est	08/08/2020	5 years	6000

Clothing Item

Ref No	Name	Demographic	Price	Description	Discount	Main-type	Sub- type	Size
1	Hooded Sweatshirt	Men	40	Hoodie in sweatshirt fabric with a soft brushed inside. Jersey-lined hood, dropped	20	Sweat Shirt	Hoodie	Large

				shoulders, long sleeves and ribbing at the cuffs and hem.				
2	Blank T-shirt	Unisex	15	Round-necked jersey T-shirt in a cotton blend.	0	Tops	T-shirt	Medium
3	Denim Jeggings	Kids	25	Jeggings in washed cotton denim with a skinny fit through the hip, thigh and leg. An easy pull-on, elasticated waist, a fake fly, fake front pockets, and open back pockets.	10	Trousers	Leggings	Small
4	Puffer Jacket	Men	35	Short puffer jacket in a quilted weave with a stand-up collar, zip down the front and welt front pockets. Loose fit with dropped shoulders, narrow elastication at the cuffs and an elasticated drawstring with cord stoppers at the hem. Lined.	15	Jacket	Puffer	X-Large
5	Sports Shorts	Men	20	Sports shorts in fabric that helps wick away moisture from your skin, keeping you comfortably dry while you move. Regular fit with four-way stretch for maximum comfort and added mobility. Elasticated waist with a concealed drawstring, diagonal side pockets and a small slit at the hem.	5	Bottoms	Shorts	Medium
6	Zip-Up Jacket	Women	25	Oversized, zip-through hoodie in sweatshirt fabric made from a cotton blend with a soft brushed inside. Drawstring hood, a zip down the front, kangaroo pockets and ribbing at the cuffs and hem.	35	Jacket	Zip-up	Small
7	Fluffy Jumper	Kids	30	Loose-fit jumper in a fluffy rib knit with a round neckline, low dropped shoulders and long sleeves.	20	Sweat shirt	Jumper	Medium
8	Suit Trousers	Men	45	Suit trousers in a stretch weave with a concealed hook-and-eye fastener and zip fly. Side pockets, welt back pockets and legs with creases. Skinny fit – a fit with slightly shorter legs that is close-fitting at the thighs,	10	Trouser	Suit	Large

				knees and ankles to create a completely fitted silhouette.				

Cosmetic Items

Cosmetic Reference Nb	Cosmetic Name	price	Cosmetic Description	Discount	Quantity	Period after opening
1	Lip Oil	10	Creamy lip oil glides like silk to achieve a uniform and eye-catching finish in a single swipe. This hydrating formula smoothes lips while delivering a wash of color and gealm you won't even notice you have on.	30	1.2 fluid ounce	12 months
2	Liquid Eyeshadow	34	The ultra-light formula glides evenly, achieving a high fixed finish. Due to its high-water content, it provides a cold effect for a pleasant and sensory application.	0	0.9 fluid ounce	12 months
3	Face brush	55	Small face brush for the application of powder, cram, or liquids n more localized areas. Rounded and precise shape.	20	null	null
4	Eyelash serum	44	Eyelash serum made of 90% of natural ingredients. The formula makes eyelashes stronger and longer, providing visible results after 4 weeks of continued use.	0	1.8 fluid ounce	6 months
5	Mascara	12	High-pigmented mascara with a formula that curls your eyelashes and achieves an adjustable effect.	50	1.2 fluid ounce	18 months
6	Cream eyeshadow	14	Creamy formula with a mousse texture, providing effortless application without the use of a brush for a delicate and a natural finish.	0	0.9 fluid ounce	12 months

7	Eyebrow gel	16	Transparent eyebrow gel. Its lightweight universal formula allows you to comb and shape your brows without leaving any residue	0	0.9 fluid ounce	18 months

Customer

Customer Email	First Name	Last Name	gender	Password	Address	Phone Nb	Credit Card Info	Date of Birth
cillainmurphy@gmail.com	Cillian	Murphy	M	98jjuT	Beirut, Ashrafieh	78999000	4000-1289-8799-0000	01/01/1969
nouraelhajj@gmail.com	Noura	El Hajj	F	onMyway12	Jbeil,	03444555	4000-1289-8799-9988	01/01/1988
pamela_ayoub@gmail.com	Pamela	Ayoub	F	pamelAAyoub	Bekaa, Mansoura	70818777	1230-1289-8799-0000	01/01/2000
samerabdou7@gmail.com	Samer	Abdou	M	SamerOnTop78	Beirut, Ein El Mrayse	76912298	4998-1289-8799-7765	01/01/1999
ghandour98@gmail.com	Mohamad	Ghandour	M	Biscuit1998	Beirut, Hamra	89999000	3000-1289-6799-9666	01/01/1998
lana_saad@gmail.com	Lana	Saad	F	Saad123Lana	Beirut, Concorde	76000999	2000-1289-8799-0000	01/01/2003
diana_ghandour@gmail.com	Diana	Ghandour	F	Ghandour0D	Beirut, Mar El Ais	70999222	4090-1289-8799-0000	01/01/1990

Returned By clothing

Clothing Reference Nb	Customer Email	Status	Return date
1	cillainmurphy@gmail.com	Exchanged	11/10/2023
2	nouraelhajj@gmail.com	Returned	14/10/2023

3	diana_ghandour@gmail.com	Rejected	13/9/2023
4	pamela_ayoub@gmail.com	Rejected	17/10/2022
5	ghandour98@gmail.com	Pending	11/10/2022
6	samerabdou7@gmail.com	Rejected	9/8/2020

Ordered By Clothing

Clothing Reference Nb	Customer Email	Status	Date of Placement	Payment Method
1	cillainmurphy@gmail.com	completed	10/10/2023	COD
2	nouraelhajj@gmail.com	completed	9/10/2023	Visa
3	diana_ghandour@gmail.com	completed	8/9/2023	Visa
4	pamela_ayoub@gmail.com	completed	7/10/2022	COD
5	ghandour98@gmail.com	completed	8/10/2022	COD
6	samerabdou7@gmail.com	completed	8/8/2023	Visa
7	lana_saad@gmail.com	processing	5/3/2023	COD

Ordered By Cosmetic

Cometic Reference Nb	Customer Email	Status	Date of Placement	Paytment Method
1	cillainmurphy@gmail.com	completed	10/10/2023	COD
2	nouraelhajj@gmail.com	completed	9/10/2023	Visa
3	nouraelhajj@gmail.com	processing	10/10/2023	COD
3	diana_ghandour@gmail.com	completed	8/9/2023	Visa
4	diana_ghandour@gmail.com	delivering	07/07/2023	Visa
4	pamela_ayoub@gmail.com	completed	7/10/2022	COD
5	ghandour98@gmail.com	completed	8/10/2022	COD
6	samerabdou7@gmail.com	completed	8/8/2023	Visa
7	lana_saad@gmail.com	delivering	5/3/2023	COD

Returned By cosmetic

Cosmetic Reference Nb	Customer Email	Status	Date_of_return
1	cillainmurphy@gmail.com	Exchanged	20/10/2023
2	nouraelhajj@gmail.com	Exchanged	10/10/2023
3	diana_ghandour@gmail.com	Refunded	11/9/2023
4	pamela_ayoub@gmail.com	Refunded	10/10/2022
5	ghandour98@gmail.com	Exchanged	11/10/2022
6	samerabdou7@gmail.com	Refunded	20/10/2023

Present In Cosmetic

Cosmetic Reference Nb	Landline	Quantity
1	01345632	100
1	01683444	89
2	01683444	200
3	01683444	77
3	05836986	8
3	09527997	100
4	05188927	90
5	05836986	25
6	009661234778	60
7	009661938224	80

Present In Clothing

Clothing Reference Nb	Landline	Quantity
1	01345632	5
2	01683444	10
3	09527997	100
4	05188927	90
5	05836986	25
6	009661234778	60
7	009661938224	80
8	09352876	40

Photos Cosmetic

Cosmetic Reference Nb	Cosmetic Photos
1	d97aa1ed849792ffe4b766239b16edbe404caa4f.png
2	7e7691fee95ff03e0360a541af450b2194102130.png
2	Hdsbvefbejhrfljkbrui47737t.png
3	25479tshai5295gvms.png
3	0ea36dfd241f67fca9958275893c4385999da00b.png
3	57223ba465a74378444e0128acd193e2d84662a6.png
4	57wiurygwhbfjywe67828628acd193e2d84662a6.png
5	75fb169ac1d18c8317e4426b51ba0e986ed7ee72png
6	Huwiubfqhbeflkjbryigewlbfywegfiubqiufhbakfsd.png
6	Ajbsfiberytwoueibriuagor7gqourvuyawv.png
7	jaeribrqvutefuhYGyFTY.png

Photos Clothing

Clothing Reference Nb	Clothing Photos
1	a911f0b10ed6ff19b5ccf0ff10bdf803ade30d32,png
1	b0daa85d081a9934c7d04e22fb53218220f558d7.png
1	Aasdisudsbdsdsdsds.png
2	e7f99025f844033b389c881ddb8e135979531558.png
3	F461230de573d530e79dae1cc5d2a3a612f5ebfa.png
3	96cd3f4c1ef26042e5349dd8abf2a37519eae615.png

4	b840d9faa6def4723868957cd509ca65ca6aa344.png
5	b6847aabadb90fbdb6a58d2893c21b69a22ff825.png
5	d60a4880ab3c11180be0e8cfac252e9efdc04b38.png
5	6c25e9af68274bc0f911a5d854d58d1f130cade4.png
6	968119979ea431833616077db69a8004c40cb35d.png
7	d2eea4c1efec156c582e7073bc9445acdc07c8d7.png
7	E7f39d6dfb109b7d2456ce99720ce7a3cd9b469f.png
8	Y378424g28fb28fv2f3r.png
8	Jfgywgf87yef8h39rnnf312.png

Color Clothing:

Clothing Reference Nb	Clothing Color
1	Black
1	White
1	Burgundy
2	Black
3	Denim Blue
3	Denim Black
4	Black
5	Dark Grey
5	Navy Blue
5	Black
6	Black
7	Cream
7	Pink
8	Black
8	Navy Blue

Color Cosmetic:

Cosmetic Reference Nb	Cosmetic Color
1	Sheer Red
2	Soft Gold
2	Purple
3	Black
3	Red
3	White
4	Transparent
5	Black
6	Cool Taupe
6	Warm Beige
7	Brown

Material:

Clothing Reference Nb	Materials
1	Cotton
1	Polyester
2	Polyester
2	Spandex
2	Rayon
3	Spandex
4	Polyester
5	Polyester
5	Cotton
6	Polyester
6	Chiffon
7	Polyamide
8	Rayon

Ingredients:

Cosmetic Reference Nb	Ingredients
1	Plant Dervid Squalene
1	Quinoa Seed Oil

2	Aqua / Water
2	Propylene Glycol
2	Talc
2	Synthetic Fluorphlogopite
3	Silicone
4	Olaaplex Peptide
4	Hyaluronic Acid
5	Castor Oil
5	Glyceryl
4	Biotin
6	Crepe
6	Ethylhexyl Palmitate
6	Isotearyl Isostearate
6	Mica
7	Beeswax

Coupon

Coupon Code	Type	Number of times used
1	30% discount on overall order	4
2	20% discount on overall order using code ‘yara’	10
3	40% discount on cosmetic order using code ‘nour’	9
4	Buy 2 get 2 free	5
5	Buy 1 get 1 free	2
6	Free shipping using code ‘elissa’	23
7	10% discount on overall order	3

Coupon Given To

Customer Email	Code
cillainmurphy@gmail.com	1

cillainmurphy@gmail.com	2
nouraelhajj@gmail.com	7
nouraelhajj@gmail.com	2
pamela_ayoub@gmail.com	3
samerabdou7@gmail.com	4
ghandour98@gmail.com	5
lana_saad@gmail.com	6
diana_ghandour@gmail.com	7

10. Implementation on PostgreSQL:

This implementation was modified according to the normalization in section 12.

10.1. Creation of All Tables and Views

```
CREATE TABLE "Clothing/Accessory item name"(
"clothing_name" character varying not null,
"clothing_price" double precision not null,
"clothing_discount" double precision,
"clothing_description" character varying,
"main_type" character varying not null,
"sub_type" character varying not null,
"demographic" character varying not null,
PRIMARY KEY ("clothing_name"),
check("clothing_discount">=0 and "clothing_discount"<1),
check(("main_type"='coat' and "sub_type" in ('coats','puffer jackets','trench coats','waistcoats')) or
  ("main_type"='jackets' and "sub_type" in ('jackets','biker','denim','leather')) or
  ("main_type"='blazers' and "sub_type" in ('blazers')) or
  ("main_type"='dresses/jumpsuits' and "sub_type" in ('midi/maxi','short','jumpsuits','long sleeve')) or
      ("main_type"='tops/bodysuits' and "sub_type" in ('long sleeve','bodysuits','crop tops','t-
shirts','knitwear')) or
  ("main_type"='shirts' and "sub_type" in ('blouses','shirts','cropped')) or
  ("main_type"='t-shirts' and "sub_type" in ('basics','long sleeve','short sleeve','tank tops','striped')) or
```

```
  ("main_type"='sweatshirts' and "sub_type" in ('basics','hoodies','zip')) or
  ("main_type"='trousers' and "sub_type" in ('wide-leg','cargo','joggers','jeans')) or
  ("main_type"='skirts/shorts' and "sub_type" in ('mini','midi','denim','shorts')) or
  ("main_type"='shoes' and "sub_type" in ('boots','ankle boots','flat shoes','high-heels','trainers','flat
sandals')) or
  ("main_type"='bags' and "sub_type" in ('crossbody','shoulder bags','backpack')) or
  ("main_type"='accessories' and "sub_type" in ('jewellery','belts','beanies','socks')) or
  ("main_type"='suits' and "sub_type" in ('formal','blazers'))
 ),
check("demographic" in ('men','women','kids','unisex'))
);
CREATE TABLE "Clothing/Accessory item"(
"clothing_RefNb" integer not null,
"clothing_name" character varying not null,
"size" character varying not null,
PRIMARY KEY ("clothing_RefNb"),
FOREIGN KEY ("clothing_name") REFERENCES "Clothing/Accessory item name"("clothing_name")
ON DELETE CASCADE ON UPDATE CASCADE,
check("size" in ('XS','S','M','L','XL','XXL','29','30','31','32','33','34','35',
 '36','37','38','39','40','41','42','43','44','45','46','47','small','medium','large',
'3-4 years','4-6 years','6-8 years','8-10 years','10-12 years','12-14 years'))
);



CREATE TABLE "Cosmetic item"(
"cosmetic_RefNb" integer not null,
"cosmetic_name" character varying not null unique,
"cosmetic_description" character varying,
"cosmetic_discount" double precision,
"cosmetic_price" double precision not null,
```

```
"quantity" double precision ,
"period_after_opening" character varying,
PRIMARY KEY ("cosmetic_RefNb"),
check("cosmetic_discount">=0 and "cosmetic_discount"<1)  );

CREATE TABLE "Customer"(
        "customer_email" varchar not null,
        "FName" varchar,
        "LName" varchar,
        "password" varchar not null,
        "gender" varchar,
        "address" varchar,
        "phonenumber" integer,
        "credit card" bigint,
        "DOB" date,
        primary key("customer_email"),
        check(length("phonenumber"::text)>6 AND length("phonenumber"::text)<15),
        check("gender" in ('F','M')),
        check(length("credit card"::text)=16),
        check("DOB"<=current_date));


      CREATE TABLE "Ordered By Clothing"(
              "clothing_RefNb" integer not null,
              "customer_email" character varying not null,
              "status" character varying not null,
              "date_of_placement" date not null,
              "payment_method" character varying not null,
              PRIMARY KEY ("clothing_RefNb","customer_email","date_of_placement"),
              FOREIGN KEY ("clothing_RefNb") references "Clothing/Accessory item"("clothing_RefNb")
```

```
        On delete cascade on update cascade,
        FOREIGN KEY ("customer_email") references "Customer"("customer_email") on delete
        Cascade on update cascade,
        check ("status" in ('order placed', 'processing', 'delivering', 'completed')),
        check ("payment_method" in ('cash','card'))
);
CREATE TABLE "Ordered By Cosmetic"(
        "cosmetic_RefNb" integer not null,
        "customer_email" character varying not null,
        "status" character varying not null,
        "date_of_placement" date not null,
        "payment_method" character varying not null,
        PRIMARY KEY ("cosmetic_RefNb","customer_email","date_of_placement"),
        FOREIGN KEY ("cosmetic_RefNb") references "Cosmetic item"("cosmetic_RefNb")
        ON DELETE cascade ON UPDATE cascade,
        FOREIGN KEY ("customer_email") references "Customer"("customer_email")
        ON DELETE cascade ON UPDATE cascade,
        check ("status" in ('order placed', 'processing', 'delivering', 'completed')),
        check ("payment_method" in ('cash','card'))
);
CREATE TABLE "Returned By Clothing"(
        "clothing_RefNb" integer not null,
        "customer_email" character varying not null,
        "status" character varying not null,
        "date_of_return" date not null,
        PRIMARY KEY ("clothing_RefNb","customer_email","date_of_return"),
        FOREIGN KEY ("clothing_RefNb") references "Clothing/Accessory item"("clothing_RefNb")
        ON DELETE cascade ON UPDATE cascade,
        FOREIGN KEY ("customer_email") references "Customer"("customer_email")
        ON DELETE cascade ON UPDATE cascade,
```

```
        check ("status" in ('returned','exchanged','rejected'))
);
CREATE TABLE "Returned By Cosmetic"(
        "cosmetic_RefNb" integer not null,
        "customer_email" character varying not null,
        "status" character varying not null,
        "date_of_return" date not null,
        PRIMARY KEY ("cosmetic_RefNb","customer_email","date_of_return"),
        FOREIGN KEY ("cosmetic_RefNb") references "Cosmetic item"("cosmetic_RefNb")
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY ("customer_email") references "Customer"("customer_email")
        ON DELETE CASCADE ON UPDATE CASCADE,
        check ("status" in ('returned','exchanged','rejected'))
);



CREATE TABLE Department (
   department_name TEXT PRIMARY KEY,
   manager_ID INT, --foreign key constraint will be added after creating Employee table
   department_landline VARCHAR(15) unique,
   email TEXT unique,
   main_office TEXT
);



CREATE TABLE Employee (
   employee_ID serial PRIMARY KEY,
   employee_email TEXT NOT NULL UNIQUE,
   salary DOUBLE PRECISION NOT NULL,
```

```
    Fname TEXT NOT NULL,
    Lname varchar NOT NULL,
    password varchar NOT NULL,
    date_of_birth DATE,
    position varchar,
    supervisor_ID INT,
    department_name TEXT REFERENCES Department(department_name)
);
ALTER TABLE Employee ADD CONSTRAINT valid_email_format check (employee_email like
'%@433.com');
ALTER TABLE Department add constraint "dept_mngr_fk" foreign key(manager_ID) references
Employee(employee_ID);
CREATE VIEW EmployeeAge AS
SELECT
    employee_ID,
    Fname,
    Lname,
    date_of_birth,
    AGE(date_of_birth) AS age
FROM Employee;
CREATE VIEW CustomerAge AS
SELECT
    "customer_email",
    "FName",
    "LName",
    "password",
            "gender",
            "address",
            "phonenumber",
            "credit card",
    AGE("DOB") AS "age"
```

```
FROM "Customer";

CREATE TABLE "Warehouse"(
"landline" bigint not null,
"Manager_ID" integer not null,
"Location" character varying,
primary key("landline"),
FOREIGN KEY ("Manager_ID") references Employee(employee_ID)
On delete cascade on update cascade,
check(length("landline"::text)>6 and length("landline"::text)<13));

CREATE TABLE "Present_in_clothing"(
"clothing_RefNb" integer not null,
"landline" bigint not null,
"quantity" integer not null,
PRIMARY KEY ("clothing_RefNb","landline"),
        FOREIGN KEY ("clothing_RefNb") references "Clothing/Accessory item"("clothing_RefNb")
On delete cascade on update cascade,
        FOREIGN KEY ("landline") references "Warehouse"("landline")
On delete cascade on update cascade,
check("quantity">=0)
);

CREATE TABLE "Present_in_cosmetics"(
"cosmetic_RefNb" integer not null,
"landline" bigint not null,
"quantity" integer not null,
PRIMARY KEY ("cosmetic_RefNb","landline"),
        FOREIGN KEY ("cosmetic_RefNb") references "Cosmetic item"("cosmetic_RefNb")
On delete cascade on update cascade,
```

```
        FOREIGN KEY ("landline") references "Warehouse"("landline")
On delete cascade on update cascade,
check("quantity">=0));



CREATE TABLE "Coupon"(
        "code" varchar,
        "type" varchar,
        "number_of_times_used" integer,
        primary key("code")
);
CREATE TABLE "Coupon_GivenTo"(
"customer_email" varchar,
"code" varchar,
primary key("customer_email","code"),
FOREIGN KEY ("customer_email") references "Customer"("customer_email")
On delete cascade on update cascade,
FOREIGN KEY ("code") references "Coupon"("code")
On delete cascade on update cascade);

CREATE TABLE "Photos Cosmetic"(
   "cosmetic_RefNb" INT NOT NULL,
   "cosmetic_photos" oid,
   FOREIGN KEY ("cosmetic_RefNb") REFERENCES "Cosmetic item"("cosmetic_RefNb")
On delete cascade on update cascade
);
CREATE TABLE "Photos Clothing"(
```

```
    "clothing_RefNb" INT NOT NULL,
    "clothing_photos" oid,
    FOREIGN KEY ("clothing_RefNb") REFERENCES "Clothing/Accessory item"("clothing_RefNb")
On delete cascade on update cascade
);
CREATE TABLE "Color Cosmetic"(
    "cosmetic_RefNb" INT NOT NULL,
    "cosmetic_color" VARCHAR(255),
    FOREIGN KEY ("cosmetic_RefNb") REFERENCES "Cosmetic item"("cosmetic_RefNb")
On delete cascade on update cascade
);
CREATE TABLE "Color Clothing"(
    "clothing_RefNb" INT NOT NULL,
    "clothing_color" VARCHAR(255),
    FOREIGN KEY ("clothing_RefNb") REFERENCES "Clothing/Accessory item"("clothing_RefNb")
On delete cascade on update cascade
);
CREATE TABLE "Material"(
    "clothing_RefNb" INT NOT NULL,
    "material" VARCHAR(255),
    FOREIGN KEY ("clothing_RefNb") REFERENCES "Clothing/Accessory item"("clothing_RefNb")
On delete cascade on update cascade
);
CREATE TABLE "Ingredients"(
    "cosmetic_RefNb" INT NOT NULL,
    "ingredients" VARCHAR(255),
    FOREIGN KEY ("cosmetic_RefNb") REFERENCES "Cosmetic item"("cosmetic_RefNb")
On delete cascade on update cascade
);
```

```
CREATE TABLE Dependent (
    employee_ID INT REFERENCES Employee(employee_ID),
    Fname VARCHAR(50) NOT NULL,
    Lname VARCHAR(50) NOT NULL,
    Date_of_birth DATE,
    relationship VARCHAR(50),
    PRIMARY KEY (employee_ID, Fname, Lname)
);

CREATE TABLE Contractor (
    contractor_name TEXT PRIMARY KEY,
    email TEXT UNIQUE,
    contractor_phone_number bigint UNIQUE,
check(length("contractor_phone_number"::text)>6 and length("contractor_phone_number"::text)<14)
);

CREATE TABLE "Has Contract With" (
    department_name TEXT REFERENCES Department(department_name),
    contractor_name TEXT REFERENCES Contractor(contractor_name),
    contract_start_date DATE,
    validity text,
    total_payment double precision,
    PRIMARY KEY (department_name, contractor_name)
);
```

10.2. Triggers

-- The following trigger is to prevent adding a return on clothing which was not ordered by a --customer before. (it removes such violating tuples without raising an assertion)

-- Note that some of these triggers could have been created as assertions, but PostgreSQL doesn't --support assertions. We would have preferred to have them as assertions.

```
Create or replace function valid_return_clothing()
returns trigger as $$
BEGIN
        delete from "Returned By Clothing" where ("clothing_RefNb","customer_email") not in
                (select "clothing_RefNb","customer_email" from "Ordered By Clothing");
        return new;
END;

$$ Language plpgsql;
create TRIGGER "valid_return_trigger_clothing" after insert on "Returned By Clothing"
for each row execute function valid_return_clothing();
```

-- The following trigger is to prevent adding a return on cosmetics which was not ordered by a --customer before.

```
Create or replace function valid_return_cosmetic()
returns trigger as $$
BEGIN
        delete from "Returned By Cosmetic" where ("cosmetic_RefNb","customer_email") not in
                (select "cosmetic_RefNb","customer_email" from "Ordered By Cosmetic");
        return new;
END;

$$ Language plpgsql;
create TRIGGER "valid_return_trigger_cosmetic" after insert on "Returned By Cosmetic"
for each row execute function valid_return_cosmetic();
```

--the following trigger is to reject returns after one month for clothing

```
Create or replace function return_clothing()
returns trigger as $$
BEGIN
        delete from "Returned By Clothing" where ("clothing_RefNb","customer_email") in
                (select "O"."clothing_RefNb","O"."customer_email" from "Ordered By Clothing" as
"O","Returned By Clothing" as "R"
                 where "O"."clothing_RefNb"="R"."clothing_RefNb"
                 and "O"."customer_email"="R"."customer_email" and
"R"."date_of_return">("O"."date_of_placement"+ interval '30 days'));
        return new;
END;

$$ Language plpgsql;
create TRIGGER "return_restriction_clothing" after insert on "Returned By Clothing"
for each row execute function return_clothing();
```

--this trigger is to reject returns after one month on cosmetics(it deletes the violating tuples --instantly)

```
Create or replace function return_cosmetics()
returns trigger as $$
BEGIN
        delete from "Returned By Cosmetic" where ("cosmetic_RefNb","customer_email") in
                (select "O"."cosmetic_RefNb","O"."customer_email" from "Ordered By Cosmetic" as
"O","Returned By Cosmetic" as "R"
                 where "O"."cosmetic_RefNb"="R"."cosmetic_RefNb"
                 and "O"."customer_email"="R"."customer_email" and
"R"."date_of_return">("O"."date_of_placement"+ interval '30 days'));
        return new;
END;

$$ Language plpgsql;
create TRIGGER "return_restriction_cosmetic" after insert on "Returned By Cosmetic"
```

```
for each row execute function return_cosmetics();
```

--the following two triggers are to remove items who have quantity=0

```
Create or replace function remove_clothing_if_0()
returns trigger as $$
BEGIN
        delete from "Present_in_clothing" where ("quantity"=0);
                return new;
END;
$$ Language plpgsql;
create TRIGGER "remove_after_quantity_0" AFTER INSERT OR UPDATE ON "Present_in_clothing"
for each row execute function remove_clothing_if_0();

Create or replace function remove_cosmetic_if_0()
returns trigger as $$
BEGIN
        delete from "Present_in_cosmetics" where ("quantity"=0);
                return new;
END;
$$ Language plpgsql;
create TRIGGER "remove_after_quantity_0" after insert or update on "Present_in_cosmetics"
for each row execute function remove_cosmetic_if_0();
```

--this is to decrease quantity by 1 after every order placed on clothing

```
CREATE OR REPLACE FUNCTION decrease_quantity_after_every_insertion()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE "Present_in_clothing"
   SET "quantity" = "quantity" - 1
```

```
   WHERE "landline"=(select "P"."landline" from "Present_in_clothing" as "P"
                                    Where "P"."clothing_RefNb"=new."clothing_RefNb"
                                    order by random() limit 1);
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
create TRIGGER "decrease_quantity_clothing" after insert on "Ordered By Clothing"
for each row execute function decrease_quantity_after_every_insertion();
```

--this is to decrease quantity by 1 after every order placed on cosmetics

```
CREATE OR REPLACE FUNCTION decrease_quantity_cosmetic()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE "Present_in_cosmetics"
   SET "quantity" = "quantity" - 1
   WHERE "landline"=(select "P"."landline" from "Present_in_cosmetics" as "P"
                                    where "P"."cosmetic_RefNb"=new."cosmetic_RefNb"
                                    order by random() limit 1);
   RETURN NEW;
END;
 $$ LANGUAGE plpgsql;
create TRIGGER "decrease_quantity_cosmetics" after insert on "Ordered By Cosmetic"
for each row execute function decrease_quantity_cosmetic();
```

10.3. Insert Queries

```
INSERT INTO "Clothing/Accessory item name"(
        clothing_name, clothing_price, clothing_discount, clothing_description, main_type,
sub_type,demographic)
        VALUES ('Hooded Sweatshirt', 40, 0.2, 'Hoodie in sweatshirt fabric with a soft brushed inside.
Jersey-lined hood, dropped shoulders, long sleeves and ribbing at the cuffs and hem', 'sweatshirts',
'hoodies', 'unisex'),
```

('Blank T-shirt', 15,0,'Round-necked jersey T-shirt in a cotton blend.' , 't-shirts', 'short sleeve', 'unisex'),

('Denim Jeggings', 25, 0.1,'Jeggings in washed cotton denim with a skinny fit through the hip, thigh and leg. An easy pull-on, elasticated waist, a fake fly, fake front pockets, and open back pockets', 'trousers', 'jeans', 'kids'),

('Puffer Jacket', 35, 0.15, 'Short puffer jacket in a quilted weave with a stand-up collar, zip down the front and welt front pockets. Loose fit with dropped shoulders, narrow elastication at the cuffs and an elasticated drawstring with cord stoppers at the hem. Lined.', 'coat', 'puffer jackets', 'men'),

('Sports Shorts', 20, 0.05,'Sports shorts in fabric that helps wick away moisture from your skin, keeping you comfortably dry while you move. Regular fit with four-way stretch for maximum ', 'skirts/shorts', 'shorts', 'men'),

('Zip-up Jacket', 25, 0.35, 'Oversized, zip-through hoodie in sweatshirt fabric made from a cotton blend with a soft brushed inside. Drawstring hood, a zip down the front, kangaroo pockets and ribbing at the cuffs and hem.', 'jackets', 'jackets', 'women'),

('Fluffy Jumper', 30, 0.2,'Loose-fit jumper in a fluffy rib knit with a round neckline, low dropped shoulders and long sleeves.' , 'sweatshirts','basics', 'kids'),

('Suit Trousers', 45, 0.1,'Suit trousers in a stretch weave with a concealed hook-and-eye fastener and zip fly. Side pockets, welt back pockets and legs with creases. Skinny fit – a fit with slightly shorter legs that is close-fitting at the thighs, knees and ankles to create a completely fitted silhouette.' , 'suits', 'formal', 'men');

INSERT INTO "Clothing/Accessory item"(

"clothing_RefNb", clothing_name,size)

VALUES (1, 'Hooded Sweatshirt', 'L'),

(2, 'Blank T-shirt', 'M'),

(3, 'Denim Jeggings', '10-12 years'),

(4, 'Puffer Jacket','XL'),

(5, 'Sports Shorts', 'M'),

(6, 'Zip-up Jacket', 'S'),

(7, 'Fluffy Jumper','6-8 years'),

(8, 'Suit Trousers', 'L');

INSERT INTO "Cosmetic item"(

"cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

VALUES (1, 'Lip Oil','Creamy lip oil glides like silk to achieve a uniform and eye-catching finish in a single swipe. This hydrating formula smoothes lips while delivering a wash of color and gealm you won’t even notice you have on.' ,0.3, 10, 1.2 ,'12M');

```
INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (2, 'Liquid Eyeshadow','The ultra-light formula glides evenly, achieving a high fixed finish. Due to its high-water content, it provides a cold effect for a pleasant and sensory application.' ,0, 34, 0.9 ,'12M');

INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (3, 'Face brush','Small face brush for the application of powder, cram, or liquids n more localized areas. Rounded and precise shape. ' ,0.2, 55, NULL ,NULL);

INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (4, 'Mascara','High-pigmented mascara with a formula that curls your eyelashes and achieves an adjustable effect' ,0.5, 12, 1.2 ,'18M');

INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (5, 'Eyelash Serum','Eyelash serum made of 90% of natural ingredients. The formula makes eyelashes stronger and longer, providing visible results after 4 weeks of continued use.' , 0, 14, 0.9, '12M');

INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (6, 'Cream eyeshadow','Creamy formula with a mousse texture, providing effortless application without the use of a brush for a delicate and a natural finish' , 0, 14, 0.9, '12M');

INSERT INTO "Cosmetic item"(

    "cosmetic_RefNb", cosmetic_name, cosmetic_description, cosmetic_discount, cosmetic_price, quantity, period_after_opening)

    VALUES (7, 'Eyebrow gel','Transparent eyebrow gel. Its lightweight universal formula allows you to comb and shape your brows without leaving any residue' , 0, 16, 0.9, '18M');
```

```
insert into "Customer"("customer_email" ,"FName" ,"LName"  ,"gender" ,"password","address" ,
                                                                    "phonenumber" ,"credit card","DOB" )
values('cillainmurphy@gmail.com','Cillian','Murphy','M','98jjuT','Beirut,Ashrafieh',78999000,400012898
7990000,'01/02/1969'),
('nouraelhajj@gmail.com','Noura','El Hajj','F','onMyway12','Jbeil',
        71444555,4000128987999988,'03/05/1988'),
('pamela_ayoub@gmail.com','Pamela','Ayoub','F','pamelAAyoub','Bekaa,Mansoura',70818777,123012898
7990000,'06/08/2000'),
('samerabdou7@gmail.com','Samer','Abdou','M','SamerOnTop78','Beirut,Ein El
Mrayse',76912298,4998128987997765,'08/09/1999'),
('ghandour98@gmail.com','Mohamad','Ghandour','M','Biscuit1998','Beirut,Hamra',89999000,3000128967
999666,'01/01/1998'),
('lana_saad@gmail.com','Lana','Saad','F','Saad123Lana','Beirut,Concorde',76000999,2000128987990000,'
07/09/2003'),
('diana_ghandour@gmail.com','Diana','Ghandour','F','Ghandour0D','Beirut,Mar El
Ais',70999222,4090128987990000,'10/01/1990');
insert into "Coupon"("code" ,     "type",  "number_of_times_used")
values(1,'30% discount on overall order',4),
(2,'20% discount on overall order using code ‘yara’',10),
(3,'40% discount on cosmetic order using code ‘nour’',9),
(4,'Buy 2 get 2 free',5),
(5,'Buy 1 get 1 free',2),
(6,'Free shipping using code ‘elissa’',23),
(7,'10% discount on overall order',3);
insert into "Coupon_GivenTo"("customer_email","code")
values('cillainmurphy@gmail.com',1),
('cillainmurphy@gmail.com',2),
('nouraelhajj@gmail.com',7),
('nouraelhajj@gmail.com',2),
('pamela_ayoub@gmail.com',3),
('samerabdou7@gmail.com',4),
('ghandour98@gmail.com',5),
```

('lana_saad@gmail.com',6),

('diana_ghandour@gmail.com',7);

INSERT INTO "Photos Cosmetic" ("cosmetic_RefNb","cosmetic_photos")

VALUES

(1,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 1.jpeg')),

(2,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image20.jpeg')),

(2,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image21.jpeg')),

(3,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 3.jpeg')),

(3,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 31.jpg')),

(3,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 32.jpg')),

(4,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 4.jpeg')),

(5,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 5.jpeg')),

(6,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 61.jpeg')),

(6,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 62.jpeg')),

(7,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image 7.jpeg'));

INSERT INTO "Photos Clothing" ("clothing_RefNb","clothing_photos")

VALUES

(1,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image1.jpeg')),

(1,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image12 (2).jpeg')),

(1,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image13 (2).jpeg')),

(2,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image2.jpeg')),

(3,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image31.jpeg')),

(3,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image32.jpeg')),

(4,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image4.jpeg')),

(5,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image51.jpeg')),

(5,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image52.jpeg')),

(5,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image53.jpeg')),

(6,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image6.jpeg')),

```
(7,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image71.jpeg')),
(7,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image72.jpeg')),
(8,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image81.jpeg')),
(8,lo_import('C:\Program Files\PostgreSQL\16\clothphotos\image82.jpeg'));


INSERT INTO "Color Clothing" ("clothing_RefNb", "clothing_color")
VALUES
    (1, 'Black'),
    (1, 'White'),
    (1, 'Burgundy'),
    (2, 'Black'),
    (3, 'Denim Blue'),
    (3, 'Denim Black'),
    (4, 'Black'),
    (5, 'Dark Grey'),
    (5, 'Navy Blue'),
    (5, 'Black'),
    (6, 'Black'),
    (7, 'Cream'),
    (7, 'Pink'),
    (8, 'Black'),
    (8, 'Navy Blue');


INSERT INTO "Color Cosmetic" ("cosmetic_RefNb", "cosmetic_color")
VALUES
    (1, 'Sheer Red'),
    (2, 'Soft Gold'),
    (2, 'Purple'),
```

```
    (3, 'Black'),
    (3, 'Red'),
    (3, 'White'),
    (4, 'Transparent'),
    (5, 'Black'),
    (6, 'Cool Taupe'),
    (6, 'Warm Beige'),
    (7, 'Brown');

INSERT INTO "Material" ("clothing_RefNb", "material")
VALUES
    (1, 'Cotton'),
    (1, 'Polyester'),
    (2, 'Polyester'),
    (2, 'Spandex'),
    (2, 'Rayon'),
    (3, 'Spandex'),
    (4, 'Polyester'),
    (5, 'Polyester'),
    (5, 'Cotton'),
    (6, 'Polyester'),
    (6, 'Chiffon'),
    (7, 'Polyamide'),
    (8, 'Rayon');

 INSERT INTO "Ingredients" ("cosmetic_RefNb", "ingredients")
VALUES
    (1, 'Plant Dervid Squalene'),
    (1, 'Quinoa Seed Oil'),
    (2, 'Aqua / Water'),
```

```
(2, 'Propylene Glycol'),
(2, 'Talc'),
(2, 'Synthetic Fluorphlogopite'),
(3, 'Silicone'),
(4, 'Olaaplex Peptide'),
(4, 'Hyaluronic Acid'),
(5, 'Castor Oil'),
(5, 'Glyceryl'),
(4, 'Biotin'),
(6, 'Crepe'),
(6, 'Ethylhexyl Palmitate'),
(6, 'Isotearyl Isostearate'),
(6, 'Mica'),
(7, 'Beeswax');

INSERT INTO "Ordered By Clothing"(
        "clothing_RefNb", customer_email, status, date_of_placement, payment_method)
        VALUES (1, 'cillainmurphy@gmail.com', 'completed', '10/10/2023', 'cash'),
        (2, 'nouraelhajj@gmail.com', 'completed', '9/10/2023', 'card'),
        (3, 'diana_ghandour@gmail.com', 'completed', '8/9/2023', 'card'),
        (4, 'pamela_ayoub@gmail.com', 'completed', '7/10/2022', 'cash'),
        (5, 'ghandour98@gmail.com', 'completed', '8/10/2023', 'cash'),
        (6, 'samerabdou7@gmail.com', 'completed', '8/8/2023', 'card'),
        (7, 'lana_saad@gmail.com', 'processing', '5/3/2023', 'cash');
INSERT into "Ordered By Cosmetic"(
        "cosmetic_RefNb", customer_email, status, date_of_placement, payment_method)
        VALUES (1, 'cillainmurphy@gmail.com', 'completed', '10/10/2023', 'cash'),
        (2, 'nouraelhajj@gmail.com', 'completed', '9/10/2023', 'card'),
        (3, 'nouraelhajj@gmail.com', 'processing', '10/10/2023', 'cash'),
        (3, 'diana_ghandour@gmail.com', 'completed', '8/9/2023', 'card'),
```

```
(4, 'diana_ghandour@gmail.com', 'delivering', '7/7/2023', 'card'),
(4, 'pamela_ayoub@gmail.com', 'completed', '7/10/2022', 'cash'),
(5, 'ghandour98@gmail.com', 'completed', '8/10/2022', 'cash'),
(6, 'samerabdou7@gmail.com', 'completed', '8/8/2023', 'card'),
(1, 'lana_saad@gmail.com', 'delivering', '5/3/2023', 'cash');

INSERT INTO "Returned By Clothing"(
"clothing_RefNb", customer_email, status, "date_of_return")
VALUES (1, 'cillainmurphy@gmail.com', 'exchanged', '10/10/2023'),
(2, 'nouraelhajj@gmail.com', 'returned', '10/10/2023'),
(3, 'diana_ghandour@gmail.com', 'returned', '9/8/2023'),
(4, 'pamela_ayoub@gmail.com', 'returned', '7/10/2022'),
(5, 'ghandour98@gmail.com', 'exchanged', '8/10/2022'),
(6, 'samerabdou7@gmail.com', 'returned', '8/8/2023');

INSERT INTO "Returned By Cosmetic"(
"cosmetic_RefNb", customer_email, status, date_of_return)
VALUES (1, 'cillainmurphy@gmail.com', 'exchanged', '10/20/2023'),
(2, 'nouraelhajj@gmail.com', 'exchanged', '10/10/2023'),
(3, 'diana_ghandour@gmail.com', 'returned', '8/9/2023'),
(4, 'pamela_ayoub@gmail.com', 'returned', '7/10/2022'),
(5, 'ghandour98@gmail.com', 'exchanged', '8/10/2022'),
(6, 'samerabdou7@gmail.com', 'returned', '8/20/2023');

INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)
VALUES ( 200123, 'Tk19@433.com', 5000, 'Tanaka', 'Miyazono', '4g^&3e', '1990-07-01',
'IT Officer', 200128, NULL);

INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)
VALUES (200124, 'Ma66@433.com', 8000, 'Mohamad', 'Abbas', '^&gsaur', '1960-09-02',
'Business Analyst', 200126, NULL);
```

```
INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES ( 200125, 'Ab26@433.com', 6000, 'Ali', 'Baba', '^&geja', '1988-06-07', 'Senior
Fashion Designer', 200130, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES ( 200126, 'Pm55@433.com', 6000, 'Piers', 'Morgan', '(*Y&dk', '1998-02-09', 'Senior
Business Analyst', 200130, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES ( 200127, 'Ae87@433.com', 5000, 'Abbas', 'Al Nouri', 'JWGgrw', '2000-08-02', 'Customer
Service Manager', 200129, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES ( 200128, 'Jw66@433.com', 7000, 'Jacinda', 'White', 'JAauwgr', '1977-10-04', 'IT
Manager', 200129, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES ( 200129, 'As32@433.com', 20000, 'Alice', 'Shields', 'OAUsew!!', '1980-04-02',
'CEO', 200130, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES (200130,

  'Jj89@433.com',20000, 'Janice', 'Jackson','7!!u=*&', '1966-01-01', 'Sales Executive',200129, NULL);


INSERT INTO Employee (employee_ID, employee_email, salary, Fname, Lname, password,
date_of_birth, position, supervisor_ID, department_name)

VALUES (200131, 'Jb00@433.com', 9000, 'Jelena', 'Brown', 'Yewug&%', '1989-08-01', 'Marketing
Officer',200129, NULL);
```

```
INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('IT', 200128, '01357261', 'it@433.com', 'Sin El Fil');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('Finance', 200126, '01746245', 'finance@433.com', 'Sin El Fil');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('Design', 200125, '01726252', 'design@433.com', 'Clemenceau st.');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('Customer Service', 200127, '01888222', 'customerservice@433.com', 'Mathaf');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('General Management', 200129, '01987567', 'management@433.com', 'Northern Metn');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('Sales', 200130, '01720000', 'sales@433.com', 'Beirut Central District');

INSERT INTO Department (department_name, manager_ID, department_landline, email, main_office)
VALUES ('Marketing', 200131, '09123123', 'marketing@433.com', 'Metn');


UPDATE Employee
SET department_name = CASE
   WHEN employee_ID=200123 THEN 'IT'
   WHEN employee_ID=200124 THEN 'Finance'
   WHEN employee_ID=200125 THEN 'Design'
         WHEN employee_ID=200126 THEN 'Finance'
```

```
        WHEN employee_ID=200127 THEN 'Customer Service'
        WHEN employee_ID=200128 THEN 'IT'
        WHEN employee_ID=200129 THEN 'General Management'
        WHEN employee_ID=200130 THEN 'Sales'
        WHEN employee_ID=200131 THEN 'Marketing'
    ELSE department_name
END;

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200124, 'Emily', 'White', '2009-03-12', 'daughter');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200128, 'William', 'Blanchard', '1935-09-21', 'father');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200126, 'Ada', 'Shields', '1960-01-01', 'mother');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200126, 'Samer', 'El Masri', '2023-01-20', 'son');
INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200125, 'Ali', 'Jackson', '2010-03-01', 'son');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200131, 'Alice', 'Brown', '2020-10-10', 'daughter');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200123, 'Kaori', 'Miyazono', '2022-04-04', 'daughter');

INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200129, 'Tim', 'Joyce', '2020-08-31', 'son');
```

```
INSERT INTO Dependent (employee_ID, Fname, Lname, Date_of_birth, relationship)
VALUES (200125, 'Susan', 'Macey', '1990-05-01', 'wife');

insert into "Warehouse" ("landline" ,"Manager_ID" ,"Location")
values (01345632,200123,'Ain El Remmaneh'),
(01683444,200124,'Dekwaneh'),
(09527997,200127,'Byblos'),
(05188927,200126,'Byblos'),
(05836986,200128,'Baabda'),
(01234778,200129,'Riyadh'),
(01938224,200131,'Jeddah'),
(09352876,200125,'Kesserwan');

insert into "Present_in_clothing"("clothing_RefNb","landline" ,"quantity")
values (1,01345632,5),
(2,01683444,10),
(3,09527997,100),
(4,05188927,90),
(5,05836986,25),
(6,01234778,60),
(7,01938224,80),
(8,09352876,40);
 insert into "Present_in_cosmetics"("cosmetic_RefNb","landline" ,"quantity")
values (1,01345632,100),
(1,01683444,89),
(2,01683444,200),
```

(3,01683444,77),

(3,05836986,8),

(3,09527997,100),

(4,05188927,90),

(5,05836986,25),

(6,01234778,60),

(7,01938224,80);

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('GASE Specialties', 'gasesp@gase.com', '05052533');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('Sbeity Contracting', 'Sbeity@sbeity.com', '09473622');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('Compass Contracting', 'compass@comp.com', '08976252');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('ENERGYTIKA SARL', 'energytika@energ.com', '01999676');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('Yamen Est', 'yamen@yamen.com', '01878888');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('Beet General Contractors', 'beet@bgc.com', '01828999');
INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('FYCO SARL', 'fyco@fyco.com', '09888777');
INSERT INTO Contractor (contractor_name, email, contractor_phone_number)
VALUES ('SMART Contractors', 'smartcontractors@smart.com', '01468999');

```
INSERT INTO Contractor (contractor_name, email, contractor_phone_number)

VALUES ('TRUST Contracting', 'trustcontracting@trust.com', '09444555');

INSERT INTO Contractor (contractor_name, email, contractor_phone_number)

VALUES ('BFD General Contractors', 'bfdcontracting@bfd.com', '009661897226');

INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('Finance', 'Compass Contracting', '2015-02-01', '10 years', 30000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('Design', 'FYCO SARL', '2020-03-03', '4 years', 10000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('Finance', 'Yamen Est', '2021-09-08', '3 years', 10000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('Sales', 'Beet General Contractors', '2023-02-01', '2 years', 4000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('IT', 'Sbeity Contracting', '2019-05-05', '6 years', 20000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('General Management', 'TRUST Contracting', '2023-01-01', '2 years', 3000.00);


INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity,
total_payment)

VALUES ('Design', 'GASE Specialties', '2018-12-12', '5 years', 9000.00);
```

INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity, total_payment)

VALUES ('General Management', 'Compass Contracting', '2022-08-31', '2 years', 8500.00);

INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity, total_payment)

VALUES ('Finance', 'SMART Contractors', '2018-01-15', '5 years', 9500.00);

INSERT INTO "Has Contract With" (department_name, contractor_name, contract_start_date, validity, total_payment)

VALUES ('Design', 'Yamen Est', '2020-08-08', '5 years', 6000.00);

10.4. Snapshots of Tables

Clothing/Accessory item name::

	clothing_name [PK] character varying	clothing_price double precision	clothing_discount double precision	clothing_description character varying		main_type character varying	sub_type character varying	demographic character varying
1	Hooded Sweatshirt	40	0.2	Hoodie in sweatshirt fabric		sweatshirts	hoodies	unisex
2	Blank T-shirt	15	0	Round-necked jersey T-shirt		t-shirts	short sleeve	unisex
3	Denim Jeggings	25	0.1	Jeggings in washed cotton		trousers	jeans	kids
4	Puffer Jacket	35	0.15	Short puffer jacket in a quil		coat	puffer jackets	men
5	Sports Shorts	20	0.05	Sports shorts in fabric that		skirts/shorts	shorts	men
6	Zip-up Jacket	25	0.35	Oversized, zip-through hooc		jackets	jackets	women
7	Fluffy Jumper	30	0.2	Loose-fit jumper in a fluffy		sweatshirts	basics	kids
8	Suit Trousers	45	0.1	Suit trousers in a stretch we	e a completely fitted sil...	suits	formal	men

Clothing/Accessory item:

	clothing_RefNb [PK] integer	clothing_name character varying	size character varying
1	1	Hooded Sweatshirt	L
2	2	Blank T-shirt	M
3	3	Denim Jeggings	10-12 years
4	4	Puffer Jacket	XL
5	5	Sports Shorts	M
6	6	Zip-up Jacket	S
7	7	Fluffy Jumper	6-8 years
8	8	Suit Trousers	L

Color clothing:

	clothing_RefNb integer	clothing_color character varying (255)
1	1	Black
2	1	White
3	1	Burgundy
4	2	Black
5	3	Denim Blue
6	3	Denim Black
7	4	Black
8	5	Dark Grey
9	5	Navy Blue
10	5	Black
11	6	Black
12	7	Cream
13	7	Pink
14	8	Black
15	8	Navy Blue

Color Cosmetic

	cosmetic_RefNb integer	cosmetic_color character varying (255)
1	1	Sheer Red
2	2	Soft Gold
3	2	Purple
4	3	Black
5	3	Red
6	3	White
7	4	Transparent
8	5	Black
9	6	Cool Taupe
10	6	Warm Beige
11	7	Brown

Cosmetic item:

	cosmetic_RefNb [PK] integer	cosmetic_name character varying	cosmetic_description character varying	cosmetic_discount double precision	cosmetic_price double precision	quantity double precision	period_after_opening character varying
1	1	Lip Oil	Creamy lip oil glides like sil	0.3	10	1.2	12M
2	2	Liquid Eyeshadow	The ultra-light formula glide	0	34	0.9	12M
3	3	Face brush	Small face brush for the ap	0.2	55	[null]	[null]
4	4	Mascara	High-pigmented mascara w	0.5	12	1.2	18M
5	5	Eyelash Serum	Eyelash serum made of 90	0	14	0.9	12M
6	6	Cream eyeshadow	Creamy formula with a mou	0	14	0.9	12M
7	7	Eyebrow gel	Transparent eyebrow gel. It	0	16	0.9	18M

Coupon:

code [PK] character varying	type character varying	number_of_times_used integer
1	30% discount on overall order	4
2	20% discount on overall order using code 'yara'	10
3	40% discount on cosmetic order using code 'nour'	9
4	Buy 2 get 2 free	5
5	Buy 1 get 1 free	2
6	Free shipping using code 'elissa'	23
7	10% discount on overall order	3

Coupon_GivenTo

customer_email [PK] character varying	code [PK] character varying
cillainmurphy@gmail.com	1
cillainmurphy@gmail.com	2
nouraelhajj@gmail.com	7
nouraelhajj@gmail.com	2
pamela_ayoub@gmail.com	3
samerabdou7@gmail.com	4
ghandour98@gmail.com	5
lana_saad@gmail.com	6
diana_ghandour@gmail.com	7

Customer:

customer_email [PK] character varying	FName character varying	LName character varying	password character varying	gender character varying	address character varying	phonenumber integer	credit card bigint	DOB date
cillainmurphy@gmail.com	Cillian	Murphy	98jjuT11	M	Beirut,Ashrafieh	78999000	4000128987990000	1969-01-
nouraelhajj@gmail.com	Noura	El Hajj	onMyway12	F	Jbeil	71444555	4000128987999988	1988-03-
pamela_ayoub@gmail.com	Pamela	Ayoub	pamelAAyoub	F	Bekaa,Mansoura	70818777	1230128987990000	2000-06-
samerabdou7@gmail.com	Samer	Abdou	SamerOnTop78	M	Beirut,Ein El Mrayse	76912298	4998128987997765	1999-08-
ghandour98@gmail.com	Mohamad	Ghandour	Biscuit1998	M	Beirut,Hamra	89999000	3000128967999666	1998-01-
lana_saad@gmail.com	Lana	Saad	Saad123Lana	F	Beirut,Concorde	76000999	2000128987990000	2003-07-
diana_ghandour@gmail.com	Diana	Ghandour	Ghandour0D	F	Beirut,Mar El Ais	70999222	4090128987990000	1990-10-

Customer view with age:

customer_email character varying	FName character varying	LName character varying	password character varying	gender character varying	address character varying	phonenumber integer	credit card bigint	age interval
cillainmurphy@gmail.com	Cillian	Murphy	98jjuT11	M	Beirut,Ashrafieh	78999000	4000128987990000	54 years
nouraelhajj@gmail.com	Noura	El Hajj	onMyway12	F	Jbeil	71444555	4000128987999988	35 years
pamela_ayoub@gmail.com	Pamela	Ayoub	pamelAAyoub	F	Bekaa,Mansoura	70818777	1230128987990000	23 years
samerabdou7@gmail.com	Samer	Abdou	SamerOnTop78	M	Beirut,Ein El Mrayse	76912298	4998128987997765	24 years
ghandour98@gmail.com	Mohamad	Ghandour	Biscuit1998	M	Beirut,Hamra	89999000	3000128967999666	25 years
lana_saad@gmail.com	Lana	Saad	Saad123Lana	F	Beirut,Concorde	76000999	2000128987990000	20 years
diana_ghandour@gmail.com	Diana	Ghandour	Ghandour0D	F	Beirut,Mar El Ais	70999222	4090128987990000	33 years

Has Contract With:

department_name [PK] text	contractor_name [PK] text	contract_start_date date	validity text	total_payment double precision
Finance	Compass Contracting	2015-02-01	10 years	30000
Design	FYCO SARL	2020-03-03	4 years	10000
Finance	Yamen Est	2021-09-08	3 years	10000
Sales	Beet General Contractors	2023-02-01	2 years	4000
IT	Sbeity Contracting	2019-05-05	6 years	20000
General Management	TRUST Contracting	2023-01-01	2 years	3000
Design	GASE Specialties	2018-12-12	5 years	9000
General Management	Compass Contracting	2022-08-31	2 years	8500
Finance	SMART Contractors	2018-01-15	5 years	9500
Design	Yamen Est	2020-08-08	5 years	6000

Ingredients:

cosmetic_RefNb integer	ingredients character varying (255)
1	Plant Dervid Squalene
1	Quinoa Seed Oil
2	Aqua / Water
2	Propylene Glycol
2	Talc
2	Synthetic Fluorphlogopite
3	Silicone
4	Olaaplex Peptide
4	Hyaluronic Acid
5	Castor Oil
5	Glyceryl
4	Biotin
6	Crepe
6	Ethylhexyl Palmitate
6	Isotearyl Isostearate
6	Mica

Material:

clothing_RefNb integer	material character varying (255)
1	Cotton
1	Polyester
2	Polyester
2	Spandex
2	Rayon
3	Spandex
4	Polyester
5	Polyester
5	Cotton
6	Polyester
6	Chiffon
7	Polyamide
8	Rayon

Ordered By Clothing:

clothing_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_placement [PK] date	payment_method character varying
1	cillainmurphy@gmail.com	completed	2023-10-10	cash
2	nouraelhajj@gmail.com	completed	2023-09-10	card
3	diana_ghandour@gmail.com	completed	2023-08-09	card
4	pamela_ayoub@gmail.com	completed	2022-07-10	cash
5	ghandour98@gmail.com	completed	2023-08-10	cash
6	samerabdou7@gmail.com	completed	2023-08-08	card
7	lana_saad@gmail.com	processing	2023-05-03	cash
3	ghandour98@gmail.com	completed	2020-12-12	cash
6	ghandour98@gmail.com	completed	2023-11-07	cash
2	diana_ghandour@gmail.com	processing	2022-10-10	card
4	diana_ghandour@gmail.com	processing	2018-09-09	card

Ordered By Cosmetic:

cosmetic_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_placement [PK] date	payment_method character varying
1	cillainmurphy@gmail.com	completed	2023-10-10	cash
2	nouraelhajj@gmail.com	completed	2023-09-10	card
3	nouraelhajj@gmail.com	processing	2023-10-10	cash
3	diana_ghandour@gmail.com	completed	2023-08-09	card
4	diana_ghandour@gmail.com	delivering	2023-07-07	card
4	pamela_ayoub@gmail.com	completed	2022-07-10	cash
5	ghandour98@gmail.com	completed	2022-08-10	cash
6	samerabdou7@gmail.com	completed	2023-08-08	card
1	lana_saad@gmail.com	delivering	2023-05-03	cash
2	ghandour98@gmail.com	completed	2022-10-10	card
4	diana_ghandour@gmail.com	processing	2021-09-09	card
3	diana_ghandour@gmail.com	processing	2019-09-09	card
3	diana_ghandour@gmail.com	processing	2018-09-09	card

Present_in_clothing:

clothing_RefNb [PK] integer	landline [PK] bigint	quantity integer
1	1345632	5
3	9527997	100
5	5836986	25
6	1234778	60
7	1938224	80
8	9352876	40
2	1683444	9
4	5188927	89

Present_in_cosmetics:

cosmetic_RefNb [PK] integer	landline [PK] bigint	quantity integer
1	1345632	100
5	5836986	25
6	1234778	60
7	1938224	80
4	5188927	89
3	5836986	7
3	9527997	99
1	1683444	88
2	1683444	199
3	1683444	75

Returned By Clothing:

clothing_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	cillainmurphy@gmail.com	exchanged	2023-10-10
2	nouraelhajj@gmail.com	returned	2023-10-10
3	diana_ghandour@gmail.com	returned	2023-09-08
4	pamela_ayoub@gmail.com	returned	2022-07-10
5	ghandour98@gmail.com	exchanged	2022-08-10
6	samerabdou7@gmail.com	returned	2023-08-08
2	ghandour98@gmail.com	rejected	2023-11-08
3	ghandour98@gmail.com	rejected	2023-11-08
1	ghandour98@gmail.com	rejected	2023-11-08
4	ghandour98@gmail.com	rejected	2023-11-08
5	ghandour98@gmail.com	rejected	2023-11-08
6	ghandour98@gmail.com	rejected	2023-11-08
7	ghandour98@gmail.com	rejected	2023-11-08

Returned By Cosmetic:

cosmetic_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	cillainmurphy@gmail.com	exchanged	2023-10-20
2	nouraelhajj@gmail.com	exchanged	2023-10-10
3	diana_ghandour@gmail.com	returned	2023-08-09
4	pamela_ayoub@gmail.com	returned	2022-07-10
5	ghandour98@gmail.com	exchanged	2022-08-10
6	samerabdou7@gmail.com	returned	2023-08-20
3	ghandour98@gmail.com	rejected	2023-11-08
4	ghandour98@gmail.com	rejected	2023-11-08
5	ghandour98@gmail.com	rejected	2023-11-08
6	ghandour98@gmail.com	rejected	2023-11-08
1	ghandour98@gmail.com	rejected	2023-11-08

Warehouse:

landline [PK] bigint	Manager_ID integer	Location character varying
1345632	200123	Ain El Remmaneh
1683444	200124	Dekwaneh
9527997	200127	Byblos
5188927	200126	Byblos
5836986	200128	Baabda
1234778	200129	Riyadh
1938224	200131	Jeddah
9352876	200125	Kesserwan

Contractor:

contractor_name [PK] text	email text	contractor_phone_number bigint
GASE Specialties	gasesp@gase.com	5052533
Sbeity Contracting	Sbeity@sbeity.com	9473622
Compass Contracting	compass@comp.com	8976252
ENERGYTIKA SARL	energytika@energ.com	1999676
Yamen Est	yamen@yamen.com	1878888
Beet General Contractors	beet@bgc.com	1828999
FYCO SARL	fyco@fyco.com	9888777
SMART Contractors	smartcontractors@smart....	1468999
TRUST Contracting	trustcontracting@trust.com	9444555
BFD General Contractors	bfdcontracting@bfd.com	9661897226

Department:

department_name [PK] text	manager_id integer	department_landline character varying (15)	email text	main_office text
IT	200128	01357261	it@433.com	Sin El Fil
Finance	200126	01746245	finance@433.com	Sin El Fil
Design	200125	01726252	design@433.com	Clemenceau st.
Customer Service	200127	01888222	customerservice@433.com	Mathaf
General Management	200129	01987567	management@433.com	Northern Metn
Sales	200130	01720000	sales@433.com	Beirut Central District
Marketing	200131	09123123	marketing@433.com	Metn

Dependent:

employee_id [PK] integer	fname [PK] character varying (50)	lname [PK] character varying (50)	date_of_birth date	relationship character varying (50)
200124	Emily	White	2009-03-12	daughter
200128	William	Blanchard	1935-09-21	father
200126	Ada	Shields	1960-01-01	mother
200126	Samer	El Masri	2023-01-20	son
200125	Ali	Jackson	2010-03-01	son
200131	Alice	Brown	2020-10-10	daughter
200123	Kaori	Miyazono	2022-04-04	daughter
200129	Tim	Joyce	2020-08-31	son
200125	Susan	Macey	1990-05-01	wife

Employee:

employee_id [PK] integer	employee_email text	salary double precision	fname text	lname character varying	password character varying	date_of_birth date	position character varying	supervisor_id integer	department_name text
200123	Tk19@433.com	5000	Tanaka	Miyazono	4g^&3e	1990-07-01	IT Officer	200128	IT
200124	Ma66@433.com	8000	Mohamad	Abbas	^&gsaur	1960-09-02	Business Analyst	200126	Finance
200125	Ab26@433.com	6000	Ali	Baba	^&geja	1988-06-07	Senior Fashion Designer	200130	Design
200126	Pm55@433.com	6000	Piers	Morgan	(*Y&dk	1998-02-09	Senior Business Analyst	200130	Finance
200127	Ae87@433.com	5000	Abbas	Al Nouri	JWGgrw	2000-08-02	Customer Service Manager	200129	Customer Service
200128	Jw66@433.com	7000	Jacinda	White	JAauwgr	1977-10-04	IT Manager	200129	IT
200129	As32@433.com	20000	Alice	Shields	OAUsew!!	1980-04-02	CEO	200130	General Manager
200130	Jj89@433.com	20000	Janice	Jackson	7!!u=*&	1966-01-01	Sales Executive	200129	Sales
200131	Jb00@433.com	9000	Jelena	Brown	Yewug&%	1989-08-01	Marketing Officer	200129	Marketing

Employee with age (view):

employee_id integer	fname text	lname character varying	date_of_birth date	age interval
200123	Tanaka	Miyazono	1990-07-01	33 years 4 mons 9 days
200124	Mohamad	Abbas	1960-09-02	63 years 2 mons 8 days
200125	Ali	Baba	1988-06-07	35 years 5 mons 3 days
200126	Piers	Morgan	1998-02-09	25 years 9 mons 1 day
200127	Abbas	Al Nouri	2000-08-02	23 years 3 mons 8 days
200128	Jacinda	White	1977-10-04	46 years 1 mon 6 days
200129	Alice	Shields	1980-04-02	43 years 7 mons 8 days
200130	Janice	Jackson	1966-01-01	57 years 10 mons 9 days
200131	Jelena	Brown	1989-08-01	34 years 3 mons 9 days

Photos Clothing:

	clothing_RefNb integer	clothing_photos oid
1	1	17109
2	1	17110
3	1	17111
4	1	17112
5	2	17113
6	3	17114
7	3	17115
8	4	17116
9	5	17117
10	5	17118
11	5	17119
12	6	17120
13	7	17121
14	7	17122
15	8	17123
16	8	17124

Photos Cosmetic:

	cosmetic_RefNb integer	cosmetic_photos oid
1	1	17138
2	2	17139
3	2	17140
4	3	17141
5	3	17142
6	3	17143
7	4	17144
8	5	17145
9	6	17146
10	6	17147
11	7	17148

11. Queries to Solve Certain Problems

11.1. Customer Preferences Analysis

Presenting users with ads and recommendations relevant to their preferences is crucial to all parties, as it improves the customers' shopping experience, and it increases FOUR3THREE's sales. This is achieved by a recommender system. However, this recommender system needs the customer's current preferences as input so it can predict what the customer might want to buy next.

A customer is said to like an item if he/she ordered that item over 30 days ago and never returned it. To obtain the list of items (**both clothing and cosmetics**) that a certain customer likes (say, the customer whose email is ghandour98@gmail.com), we use the below query.

This query obtains all items that were ordered by the customer with email ghandour98@gmail.com over 30 days ago, *except* for the items that he returned. The UNION operator is used to combine the results of clothing/accessory items and cosmetics items.

```
SELECT "clothing_name" as " Item"  FROM "Clothing/Accessory item"
WHERE "clothing_RefNb" in ( (SELECT "clothing_RefNb" from "Ordered By Clothing"
                            where "customer_email"='ghandour98@gmail.com' and
                            "date_of_placement" < NOW()- INTERVAL '30 days')

                            EXCEPT

                            (SELECT "O"."clothing_RefNb"
                        from "Ordered By Clothing" as "O", "Returned By Clothing" as "R"
                            where "O"."clothing_RefNb"="R"."clothing_RefNb" and
                            "O"."customer_email"="R"."customer_email"
                            and "O"."customer_email"='ghandour98@gmail.com')
                            )
UNION
SELECT "cosmetic_name" as " Item"  FROM "Cosmetic item"
WHERE "cosmetic_RefNb" in ( (SELECT "cosmetic_RefNb" from "Ordered By Cosmetic"
                            where "customer_email"='ghandour98@gmail.com' and
                            "date_of_placement" < NOW()- INTERVAL '30 days')

                            EXCEPT

                            (SELECT "O"."cosmetic_RefNb"
                            from "Ordered By Cosmetic" as "O", "Returned By Cosmetic" as
"R"
                            where "O"."cosmetic_RefNb"="R"."cosmetic_RefNb" and
                            "O"."customer_email"="R"."customer_email"
and "O"."customer_email"='ghandour98@gmail.com')
                            );
```

The following output is obtained:

Data Output | Messages | Notifications

	Item character varying
1	Denim Jeggings
2	Liquid Eyeshadow

Note that these are all items that ghandour98 ordered:

	clothing_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_placement [PK] date	payment_method character varying
1	5	ghandour98@gmail.com	completed	2023-08-10	cash
2	3	ghandour98@gmail.com	completed	2020-12-12	cash
3	6	ghandour98@gmail.com	completed	2023-11-07	cash

	cosmetic_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_placement [PK] date	payment_method character varying
1	5	ghandour98@gmail.com	completed	2022-08-10	cash
2	2	ghandour98@gmail.com	completed	2022-10-10	card

And these are all items he returned:

	cosmetic_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	5	ghandour98@gmail.com	exchanged	2022-08-10

	clothing_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	5	ghandour98@gmail.com	exchanged	2022-08-10

11.2. Flagging Certain Customers

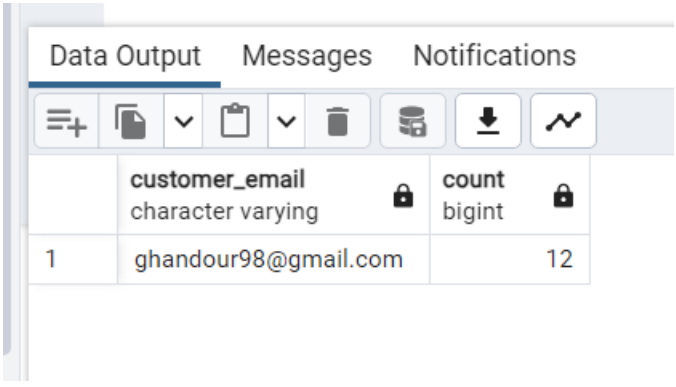
FOUR3THREE allows returns only if the item is damaged. In that case, the customer can get an exchange or refund. Otherwise, the return request is rejected. The company's return policy states that if more than 10 returns are rejected in the same month, the user is to be flagged. This prevents customers from abusing the return policy.

For this reason, FOUR3THREE needs the list of customers who had more than 10 rejected returns in the past 30 days. This is achieved by the below query.

First, all returns (both clothing and cosmetics) are combined into one table (All_Returns) which will be used for this query. UNION ALL is used because duplicates must be retained if there are rows that have the same reference number, date of return, status, and customer email. This might happen because the reference number of a cosmetic item might be equal to that of a clothing item. Then, the customers who have more than 10 rejected returns in the past 30 days are selected from "All_Returns".

```
WITH "All_Returns"("customer_email","item_refNb","date_of_return","status")  AS (

    (SELECT "customer_email","clothing_RefNb","date_of_return","status" from
    "Returned By Clothing")

    UNION ALL

    (SELECT "customer_email","cosmetic_RefNb","date_of_return","status" from
    "Returned By Cosmetic")

    )

SELECT "customer_email", count(*) from "All_Returns"

WHERE "date_of_return">NOW() -INTERVAL '30 days' and "status"='rejected'

GROUP BY "customer_email" HAVING count(*)>10;
```

The following output is obtained:



Note that these are the items that ghandour returned:

	clothing_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	5	ghandour98@gmail.com	exchanged	2022-08-10
2	2	ghandour98@gmail.com	rejected	2023-11-08
3	3	ghandour98@gmail.com	rejected	2023-11-08
4	1	ghandour98@gmail.com	rejected	2023-11-08
5	4	ghandour98@gmail.com	rejected	2023-11-08
6	5	ghandour98@gmail.com	rejected	2023-11-08
7	6	ghandour98@gmail.com	rejected	2023-11-08
8	7	ghandour98@gmail.com	rejected	2023-11-08

	cosmetic_RefNb [PK] integer	customer_email [PK] character varying	status character varying	date_of_return [PK] date
1	5	ghandour98@gmail.com	exchanged	2022-08-10
2	3	ghandour98@gmail.com	rejected	2023-11-08
3	4	ghandour98@gmail.com	rejected	2023-11-08
4	5	ghandour98@gmail.com	rejected	2023-11-08
5	6	ghandour98@gmail.com	rejected	2023-11-08
6	1	ghandour98@gmail.com	rejected	2023-11-08

11.3. Local Items

When users visit FOUR3THREE, our primary goal is to empower them with the ability to filter and discover nearby items conveniently housed in warehouses closest to them. This feature not only guarantees a swifter and more efficient delivery process but also actively contributes to the reduction of overall shipping costs, enhancing the cost-effectiveness of their shopping experience.

This query shows the Clothing/Accessory items and Cosmetic items that are found in warehouses that have the same location as a given customer (say, the customer “Hamza Chami”), so that the customer has the flexibility to handpick the items which are closest to them enabling them to effortlessly discover the availability of desired items in that location. It then combines results from “Present_in_clothing” and “Present_in_cosmetics” tables using the “UNION ALL” to merge the results while maintaining duplicates because a clothing item and a cosmetic item may have the same reference number.

```
SELECT PC."clothing_RefNb" AS item_RefNb,
  'Clothing/Accessory item' AS item_type,
  W."Location" AS warehouse_location
```

```
FROM "Present_in_clothing" AS PC
JOIN "Warehouse" AS W ON PC."landline" = W."landline"
 WHERE W."Location" like (select "address" from "Customer" where
                          "customer_email"='hamzachami@gmail.com')
UNION ALL
SELECT PC."cosmetic_RefNb" AS item_RefNb,
  'Cosmetic item' AS item_type,
  W."Location" AS warehouse_location
FROM "Present_in_cosmetics" AS PC
JOIN "Warehouse" AS W ON PC."landline" = W."landline"
WHERE W."Location" like (select "address" from "Customer" where
                          "customer_email"='hamzachami@gmail.com');
```

In this query we used the location where Hamza Chami lives (Baabda) to serve as an example to show the items that are found in the “Baabda” warehouse.

This is the customer’s info, we can see that he lives in ‘Baabda’

	customer_email [PK] character varying	FName character varying	LName character varying	password character varying	gender character varying	address character varying	phonenumber integer	credit bigint
1	hamzachami@gmail.com	Hamza	Chami	AsAs0987	M	Baabda	71829372	4000

This is the output of the query:

	item_refnb integer	item_type text	warehouse_location character varying
1	5	Clothing/Accessory item	Baabda
2	3	Cosmetic item	Baabda
3	5	Cosmetic item	Baabda

11.4. Employee Info

FOUR3THREE cares about knowing the critical insights about its workforce and its structure. Analyzing the maximum salary, average salaries and workforce distribution allows for strategic resource/workforce allocation, which is done by the general management department. In short, the management department needs a query that identifies the highest paid employee and their salary, the average salary, and the number of employees within every department.

This query first uses With Ranking As to first select the employee information from Employee table and uses the RANK () function which assigns a rank to each employee within their department based on their salaries in DESC order. This results in columns such as employee_ID, Fname, Lname, salary, department_name and a new column which is salary_rank. The next SELECT statement then gets this information and joins it with a subquery that calculates the average salary and number of employees of each department based on the column department_name. The WHERE statement then filters the result to include only the top-earning employee of each department where their salary_rank =1.

```
WITH Ranking AS (
    SELECT e.employee_ID, e.Fname, e.Lname, e.salary, e.department_name, RANK() OVER (PARTITION BY e.department_name ORDER BY e.salary DESC) as salary_rank
    FROM Employee AS e
)

SELECT r.employee_ID, r.Fname, r.Lname, r.salary, r.department_name, e.num_employees, e.avg_salary, r.salary_rank
FROM Ranking AS r
JOIN (
    SELECT department_name, COUNT(employee_ID) as num_employees, AVG(salary) as avg_salary
    FROM Employee
    GROUP BY department_name
) e ON r.department_name = e.department_name
WHERE r.salary_rank = 1
```

This is the output:

	employee_id [PK] integer	fname text	lname character varying	salary double precision	department_name text	num_employees bigint	avg_salary double precision	salary_rank bigint
1	200127	Abbas	Al Nouri	5500	Customer Service	2	4250	1
2	200125	Ali	Baba	6600.000000000001	Design	2	4886	1
3	200124	Mohamad	Abbas	8000	Finance	4	4722.25	1
4	200129	Alice	Shields	22000	General Management	2	12193	1
5	200128	Jacinda	White	7700.000000000001	IT	4	4993.25	1
6	200131	Jelena	Brown	9900	Marketing	3	4799.333333333333	1
7	200130	Janice	Jackson	20000	Sales	2	10624.5	1

These are all the employees:

	employee_id [PK] integer	employee_email text	salary double precision	fname text	lname character varying	password character varying	date_of_birth date	position character varying	supervisor_id integer	departm text
1	10	emp1@433.com	1185	Alice	Johnson	pass123	1992-03-10	Analyst	[null]	Finance
2	11	emp2@433.com	3888	Bob	Miller	secretword	1985-07-18	Coordinator	[null]	IT
3	12	emp3@433.com	1246	Charlie	Brown	securepass	1989-11-25	Associate	[null]	Marketi
4	13	emp4@433.com	1249	Diana	Taylor	password456	1994-01-05	Assistant	[null]	Sales
5	14	emp5@433.com	3000	Ethan	Wright	letmein	1987-06-30	Coordinator	[null]	Custom
6	15	emp6@433.com	3172	Fiona	Clark	access123	1991-09-15	Analyst	[null]	Design
7	16	emp7@433.com	2386	George	Harris	adminpass	1993-04-20	Assistant	[null]	General
8	17	emp8@433.com	3104	Helen	King	mypassword	1986-12-12	Coordinator	[null]	Finance
9	18	emp9@433.com	3385	Ian	Moore	letmein456	1990-02-08	Analyst	[null]	IT
10	19	emp10@433.com	3252	Jack	Baker	secure123	1988-08-05	Associate	[null]	Marketi
11	200123	Tk19@433.com	5000	Tanaka	Miyazono	4g^&3e	1990-07-01	IT Officer	200128	IT
12	200124	Ma66@433.com	8000	Mohamad	Abbas	^&gsaur	1960-09-02	Business Analyst	200126	Finance
13	200125	Ab26@433.com	6600.000000000001	Ali	Baba	^&geja	1988-06-07	Senior Fashion Designer	200130	Design
14	200126	Pm55@433.com	6600.000000000001	Piers	Morgan	(*Y&dk	1998-02-09	Senior Business Analyst	200130	Finance
15	200127	Ae87@433.com	5500	Abbas	Al Nouri	JWGgrw	2000-08-02	Customer Service Manager	200129	Custom
16	200128	Jw66@433.com	7700.000000000001	Jacinda	White	JAauwgr	1977-10-04	IT Manager	200129	IT
17	200129	As32@433.com	22000	Alice	Shields	OAUsew!!	1980-04-02	CEO	200130	General
18	200130	Jj89@433.com	20000	Janice	Jackson	7!!u=*&	1966-01-01	Sales Executive	200129	Sales
19	200131	Jb00@433.com	9900	Jelena	Brown	Yewuq&%	1989-08-01	Marketing Officer	200129	Marketi

11.5. Welcome Coupon

FOUR3THREE believes in extending a warm welcome to new users, and what better way to do so than by offering a generous 15% discount as a token of appreciation for choosing our platform for the first time. This exclusive coupon not only encourages users to experience the diverse range of products and seamless shopping process on FOUR3THREE but also serves as an incentive to build a lasting connection with our community. By providing this enticing discount, we aim to create a positive and rewarding introduction to the FOUR3THREE shopping experience, fostering a sense of value and satisfaction from the very first order.

This query checks for each customer whether they have ordered anything by checking the “ordered by clothing” table which is the table for orders of clothing/accessory items and the ordered by cosmetics table which is the table for orders of cosmetic items, and if the user’s email is not found in both of these tables which was checked by “WHERE NOT EXISTS” then that means that they have never ordered, and they are given a Welcome Coupon.

```
INSERT INTO "Coupon" ("code", "type", "number_of_times_used")
VALUES('WELCOME', '15% discount on all orders (Welcome Coupon)', 0);
INSERT INTO "Coupon_GivenTo" ("customer_email", "code")
SELECT C."customer_email", 'WELCOME'
FROM "Customer" C
WHERE NOT EXISTS (
```

```
    SELECT 1
    FROM "Ordered By Clothing" OC
    WHERE OC."customer_email" = C."customer_email"
) AND NOT EXISTS (
    SELECT 1
    FROM "Ordered By Cosmetic" OCC
    WHERE OCC."customer_email" = C."customer_email"
);
```

First, create the Welcome Coupon with the code WELCOME and type of 15% discount on all orders.

	code [PK] character varying	type character varying	number_of_times_used integer
1	1	30% discount on overall order	4
2	2	20% discount on overall order using code 'yara'	10
3	3	40% discount on cosmetic order using code 'nour'	9
4	4	Buy 2 get 2 free	5
5	5	Buy 1 get 1 free	2
6	6	Free shipping using code 'elissa'	23
7	7	10% discount on overall order	3
8	WELCOME	15% discount on all orders (Welcome Coupon)	0

Ali Hussein is the only new customer who has never ordered an item yet. This is the result of executing the query:

	customer_email [PK] character varying	code [PK] character varying
1	alihussein@gmail.com	WELCOME
2	cillainmurphy@gmail.com	1
3	cillainmurphy@gmail.com	2
4	diana_ghandour@gmail.com	7
5	ghandour98@gmail.com	5
6	lana_saad@gmail.com	6
7	nouraelhajj@gmail.com	2
8	nouraelhajj@gmail.com	7
9	pamela_ayoub@gmail.com	3
10	samerabdou7@gmail.com	4

11.6. Discount on least ordered items:

To increase the profit of the company and attract more customers, a discount on the least ordered items will be applied to encourage more customers to buy them. First the least ordered items will be chosen and on the least ordered one, a 75% discount will be applied. As for the second least ordered item, a 50% discount will be applied. And finally, for the 3rd least ordered item, a 25% discount will be applied. Note that the prices in “Cosmetic item” and “Clothing item” tables are the *discounted* prices according to the discount present in column “cosmetic_discount” or “clothing_discount”. Thus, original prices are retrieved using old price= new price/(1-discount). Then, the new discounted price is calculated by multiplying the original price by (1- new discount).

11.6.1. Discount on Cosmetic table:

```
UPDATE "Cosmetic item" set "cosmetic_price"=
CASE
        WHEN "cosmetic_RefNb" in
            (select "cosmetic_RefNb" from "Ordered By Cosmetic"
            group by "cosmetic_RefNb"
            order by count("cosmetic_RefNb") asc limit 1)
```

```
                    THEN ("cosmetic_price"/(1-"cosmetic_discount"))*0.25
                    WHEN "cosmetic_RefNb" in
                        (select "cosmetic_RefNb" from "Ordered By Cosmetic"
                        group by "cosmetic_RefNb"
                        order by count("cosmetic_RefNb") asc limit 1 offset 1)
                    THEN ("cosmetic_price"/(1-"cosmetic_discount"))*0.5
                    WHEN "cosmetic_RefNb" in
                        (select "cosmetic_RefNb"
                        from "Ordered By Cosmetic"group by "cosmetic_RefNb"
                        order by count("cosmetic_RefNb") asc limit 1 offset 2)
                    THEN ("cosmetic_price"/(1-"cosmetic_discount"))*0.75
                    ELSE "cosmetic_price"
END,
"cosmetic_discount"=
CASE
                    WHEN "cosmetic_RefNb" in
                        (select "cosmetic_RefNb" from "Ordered By Cosmetic"
                        group by "cosmetic_RefNb"
                        order by count("cosmetic_RefNb") asc limit 1)
                    THEN 0.75
                    WHEN "cosmetic_RefNb" in
                        (select "cosmetic_RefNb" from "Ordered By Cosmetic"
                        group by "cosmetic_RefNb"
                        order by count("cosmetic_RefNb") asc limit 1 offset 1)
                    THEN 0.5
                    WHEN "cosmetic_RefNb" in
                        (select "cosmetic_RefNb"
                        from "Ordered By Cosmetic"group by "cosmetic_RefNb"
```

order by count("cosmetic_RefNb") asc limit 1 offset 2)

THEN 0.25

ELSE "cosmetic_discount"

END;

Prices before updating:

	cosmetic_RefNb [PK] integer	cosmetic_name character varying	cosmetic_price double precision	cosmetic_discount double precision
1	1	Lip Oil	10	0.3
2	3	Face brush	55	0.2
3	7	Eyebrow gel	16	0
4	2	Liquid Eyeshadow	40	0.4
5	4	Mascara	45	0.4
6	5	Eyelash Serum	14	0.3
7	6	Cream eyeshadow	20	0.3

Prices after updating:

	cosmetic_RefNb [PK] integer	cosmetic_name character varying	cosmetic_price double precision	cosmetic_discount double precision
1	1	Lip Oil	10	0.3
2	3	Face brush	55	0.2
3	7	Eyebrow gel	16	0
4	4	Mascara	45	0.4
5	5	Eyelash Serum	5	0.75
6	6	Cream eyeshadow	14.285714285714286	0.5
7	2	Liquid Eyeshadow	50	0.25

11.6.2. Discount on Clothing table:

--note: this is the same query as above, but for clothing.

UPDATE "Clothing/Accessory item name" set "clothing_price"=

CASE

WHEN "clothing_name" in

```
        (select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
                select "clothing_RefNb" from "Ordered By Clothing"
                group by "clothing_RefNb"
                order by count("clothing_RefNb") asc limit 1))
        THEN ("clothing_price"/(1-"clothing_discount"))*0.25
        WHEN"clothing_name" in
        (select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
                select "clothing_RefNb" from "Ordered By Clothing"
                group by "clothing_RefNb"
                order by count("clothing_RefNb") asc limit 1 offset 1))
        THEN ("clothing_price"/(1-"clothing_discount"))*0.5
        WHEN "clothing_name" in
        (select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
                select "clothing_RefNb" from "Ordered By Clothing"
                group by "clothing_RefNb"
                order by count("clothing_RefNb") asc limit 1 offset 2))
        THEN ("clothing_price"/(1-"clothing_discount"))*0.75
        ELSE "clothing_price"
END,
"clothing_discount"=
CASE
        WHEN "clothing_name" in
        (select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
                select "clothing_RefNb" from "Ordered By Clothing"
                group by "clothing_RefNb"
                order by count("clothing_RefNb") asc limit 1))
        THEN 0.75
        WHEN "clothing_name" in
```

```
(select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
        select "clothing_RefNb" from "Ordered By Clothing"
        group by "clothing_RefNb"
        order by count("clothing_RefNb") asc limit 1 offset 1))
THEN 0.5
WHEN "clothing_name" in
(select "clothing_name" from "Clothing/Accessory item" where "clothing_RefNb" in(
        select "clothing_RefNb" from "Ordered By Clothing"
        group by "clothing_RefNb"
        order by count("clothing_RefNb") asc limit 1 offset 2))
THEN 0.25
ELSE "clothing_discount"
END;
```

Prices before updating:

	clothing_RefNb [PK] integer	clothing_price double precision	clothing_discount double precision
1	2	15	0
2	3	25	0.1
3	4	35	0.15
4	6	25	0.3
5	8	45	0.1
6	5	25	0.3
7	7	30	0.3
8	1	45	0.3

Prices after updating:

	clothing_RefNb [PK] integer	clothing_price double precision	clothing_discount double precision
1	2	15	0
2	3	25	0.1
3	4	35	0.15
4	6	25	0.3
5	8	45	0.1
6	5	8.928571428571429	0.75
7	7	21.42857142857143	0.5
8	1	23.624999999999996	0.25

11.7. Managers that are 64 years old:

It is crucial that when a manager becomes 64 years old and needs to leave for retirement, a department does not lack a manager, so in the following query, assuming only 1 employee and he/she is a manager is going to retirement, it updates department table and sets the manager of the department to the oldest supervisor in the same department. Also, its assumed here that the managers have their supervisors the CEO which is of ID 200129, and the manager is the supervisor of the department.

On a side note, for the above assumptions to be true, these insertions and updates were required for testing purposes:

update employee set date_of_birth='1959-10-04' where employee_id=200128;

--making the manager of the IT department 64 years old

insert into employee(employee_id,employee_email,salary,fname,lname,password, date_of_birth,position,supervisor_id,department_name) values (200232,'Lm32@433.com',2000,'Lana','Michelson','lana_mic90','1962-10-9','IT executive',200128,'IT');

--inserting a supervisor of the IT department that is old

update employee set supervisor_id=200232 where employee_id=200123;

--setting the new inserted employee as the supervisor of the employees in the IT department

Query:

update department set

```
manager_id=(select supervisor_id from employee where
supervisor_id =(SELECT e1.supervisor_id
FROM employee e1
JOIN department dep ON e1.department_name = dep.department_name
where dep.department_name=(select department_name from department where
manager_id in (
                SELECT manager_id
    FROM department dep
                join employee e1 on e1.employee_id=dep.manager_id
    WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '63 years'))
GROUP BY e1.supervisor_id
ORDER BY MAX(e1.date_of_birth) desc
limit 1))
```

--the above statements is to select the oldest supervisor that works in the same department as the manager who turned 64 years old

```
where manager_id in (SELECT manager_id
    FROM department dep
      join employee e1 on e1.employee_id=dep.manager_id
    WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '63 years');
```

--then we are selecting the manager that is 64 years old to replace him with the oldest supervisor

```
update employee set position=(SELECT position
    FROM employee
    WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '64 years'),
            supervisor_id=200129
where employee_id=(select manager_id from department
             where department_name=(SELECT department_name
    FROM employee
```

```
WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '64 years'));
```

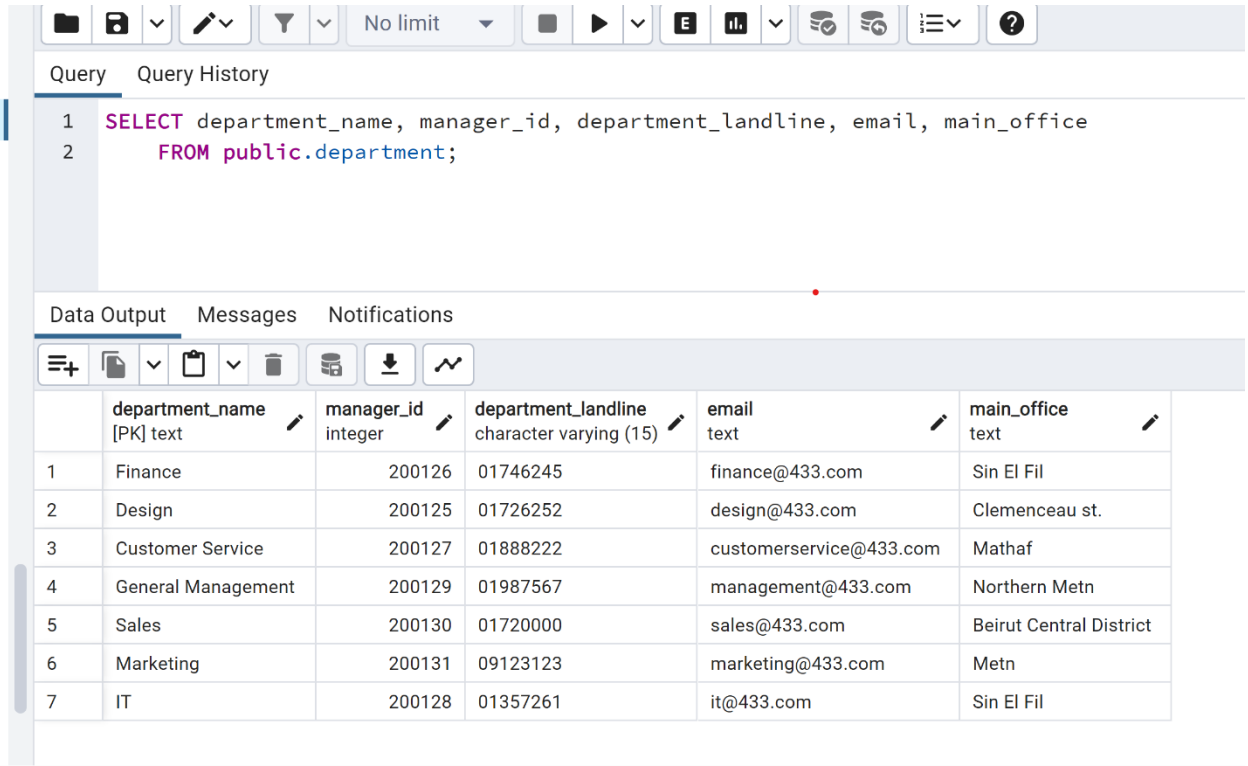
--after setting the manager of the IT department to the oldest supervisor, we are changing the position of the oldest supervisor to be the same position as the ex-manager that is 64 and setting his supervisor as the CEO because as stated before the CEO is the supervisor of all managers

```
delete from dependent where employee_id=(SELECT employee_id
    FROM employee
    WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '64 years');
```

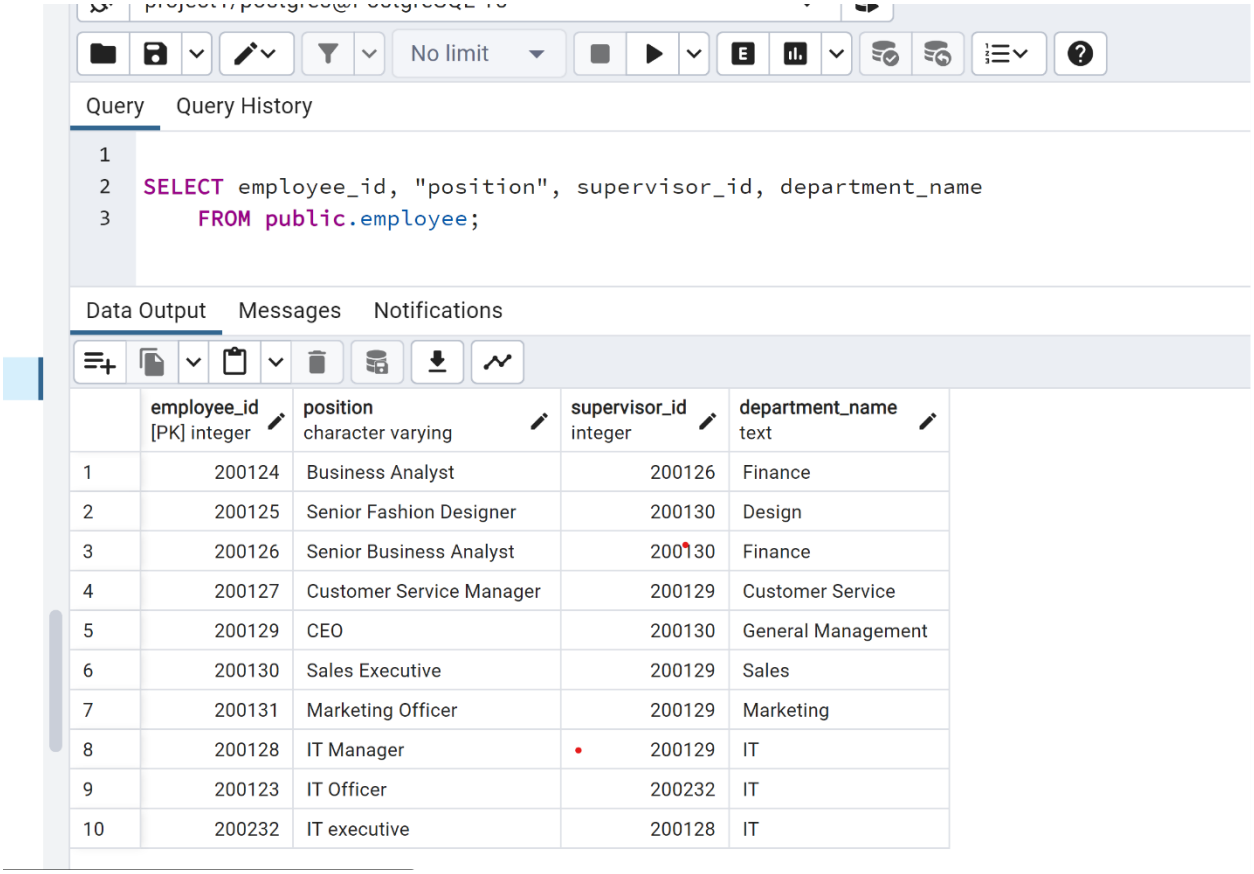
--deleting the dependents of the employee whos going to retirement to be able to delete him from employee table

```
delete from employee where employee_id=(SELECT employee_id
    FROM employee
    WHERE AGE(CURRENT_DATE, date_of_birth) > INTERVAL '64 years')
```

Before executing:



	department_name [PK] text	manager_id integer	department_landline character varying (15)	email text	main_office text
1	Finance	200126	01746245	finance@433.com	Sin El Fil
2	Design	200125	01726252	design@433.com	Clemenceau st.
3	Customer Service	200127	01888222	customerservice@433.com	Mathaf
4	General Management	200129	01987567	management@433.com	Northern Metn
5	Sales	200130	01720000	sales@433.com	Beirut Central District
6	Marketing	200131	09123123	marketing@433.com	Metn
7	IT	200128	01357261	it@433.com	Sin El Fil



No limit

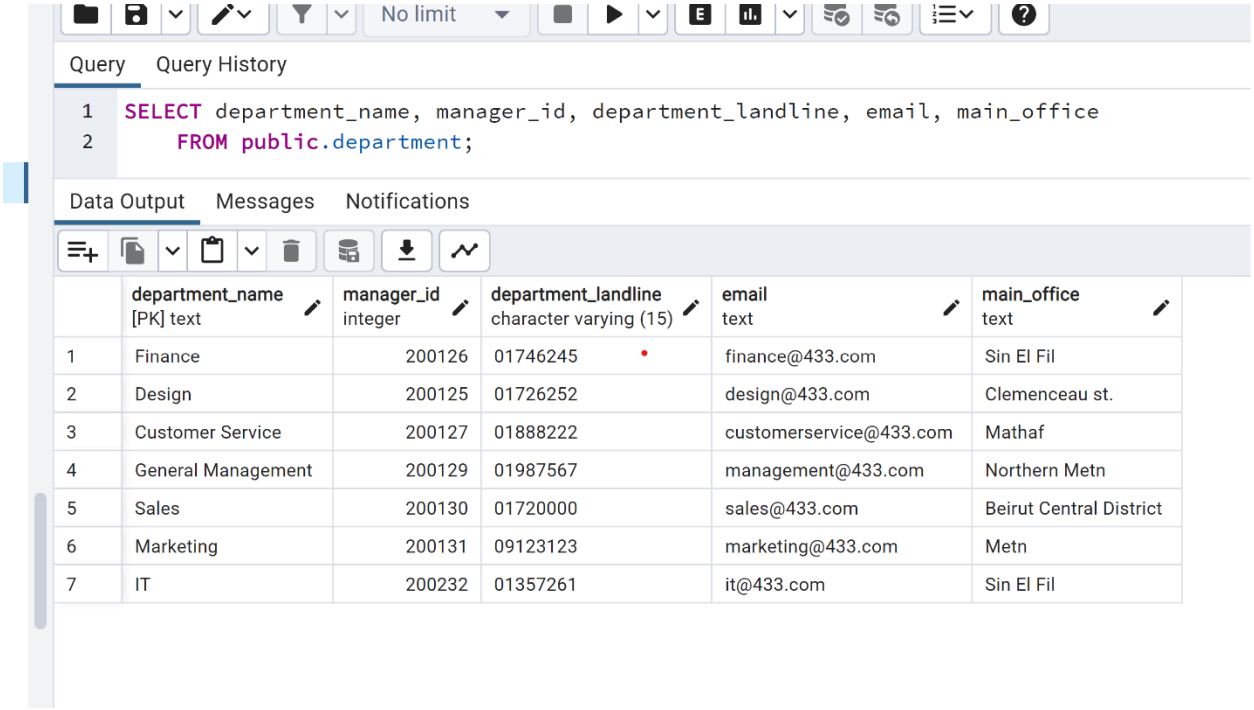
Query | Query History

```
1
2  SELECT employee_id, "position", supervisor_id, department_name
3      FROM public.employee;
```

Data Output | Messages | Notifications

	employee_id [PK] integer	position character varying	supervisor_id integer	department_name text
1	200124	Business Analyst	200126	Finance
2	200125	Senior Fashion Designer	200130	Design
3	200126	Senior Business Analyst	200130	Finance
4	200127	Customer Service Manager	200129	Customer Service
5	200129	CEO	200130	General Management
6	200130	Sales Executive	200129	Sales
7	200131	Marketing Officer	200129	Marketing
8	200128	IT Manager	200129	IT
9	200123	IT Officer	200232	IT
10	200232	IT executive	200128	IT

After executing:



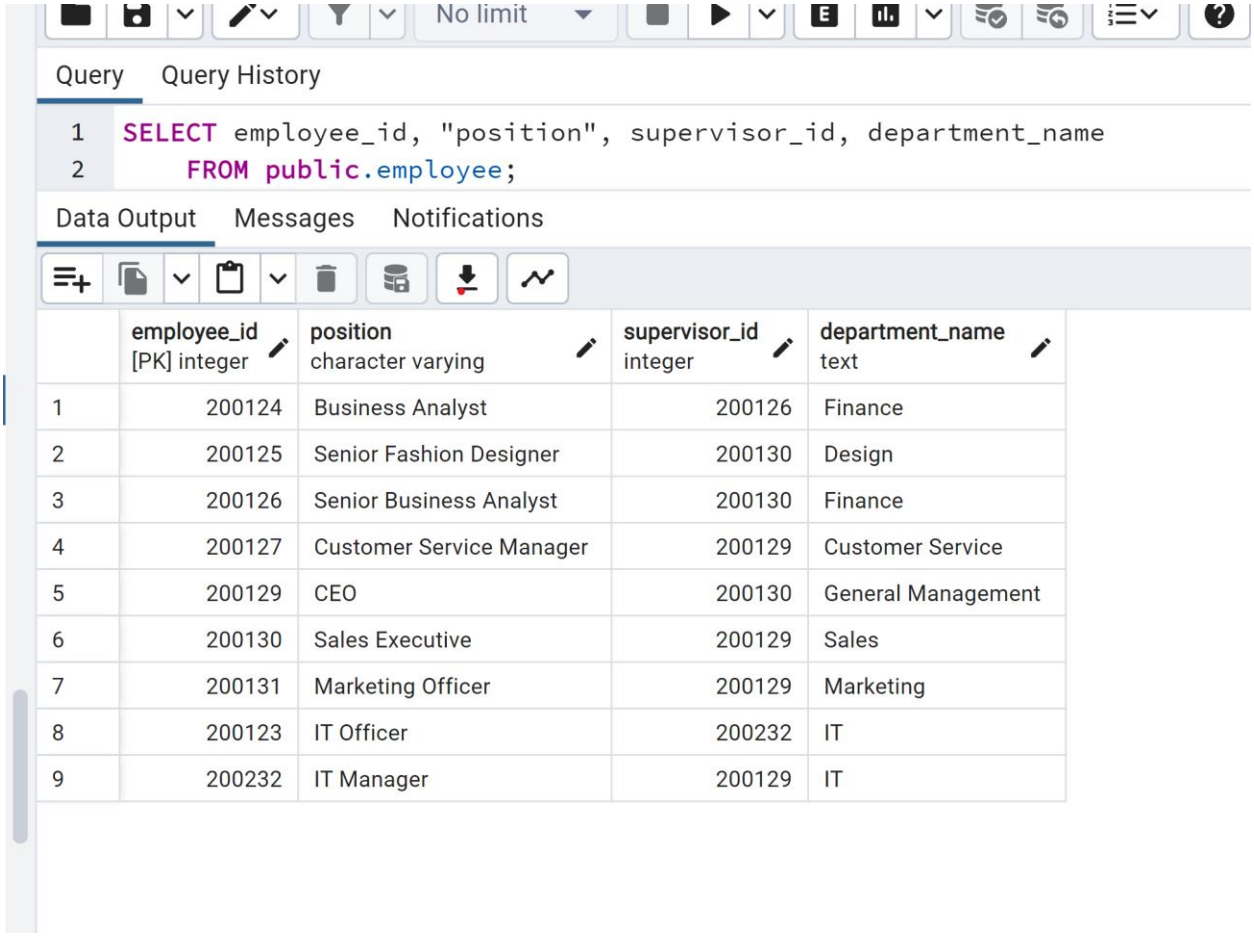
No limit

Query | Query History

```
1  SELECT department_name, manager_id, department_landline, email, main_office
2      FROM public.department;
```

Data Output | Messages | Notifications

	department_name [PK] text	manager_id integer	department_landline character varying (15)	email text	main_office text
1	Finance	200126	01746245	finance@433.com	Sin El Fil
2	Design	200125	01726252	design@433.com	Clemenceau st.
3	Customer Service	200127	01888222	customerservice@433.com	Mathaf
4	General Management	200129	01987567	management@433.com	Northern Metn
5	Sales	200130	01720000	sales@433.com	Beirut Central District
6	Marketing	200131	09123123	marketing@433.com	Metn
7	IT	200232	01357261	it@433.com	Sin El Fil



	employee_id [PK] integer	position character varying	supervisor_id integer	department_name text
1	200124	Business Analyst	200126	Finance
2	200125	Senior Fashion Designer	200130	Design
3	200126	Senior Business Analyst	200130	Finance
4	200127	Customer Service Manager	200129	Customer Service
5	200129	CEO	200130	General Management
6	200130	Sales Executive	200129	Sales
7	200131	Marketing Officer	200129	Marketing
8	200123	IT Officer	200232	IT
9	200232	IT Manager	200129	IT

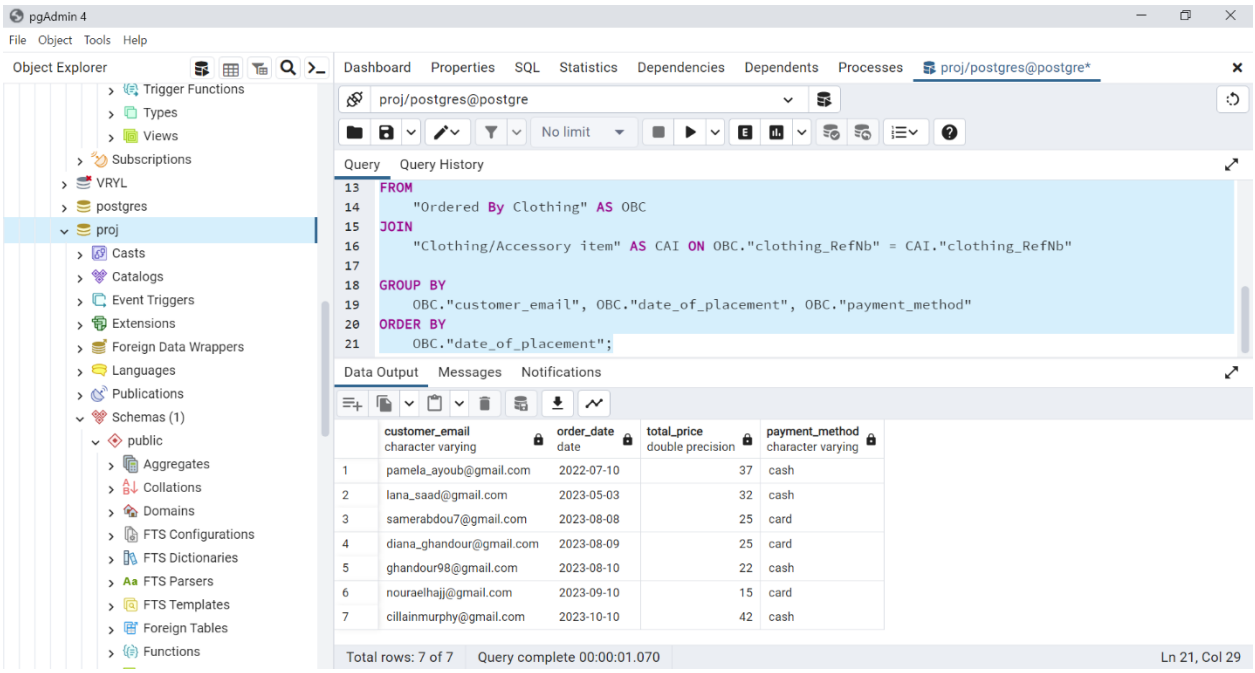
11.8. Bill for Orders:

Calculating the bill is a crucial part of placing an order. An order includes all items that were ordered at the same time (in other words, on the same date). This query selects all orders made on the same date and provides the corresponding bill with the customer's email, totalPrice, order date, and the payment method used. If cash is the corresponding payment method then 2$ are automatically added to account for the delivery fee. This is a necessary implementation to track shop activity and give a general overview on sales.

```
SELECT
    OBC."customer_email",
    OBC."date_of_placement" AS order_date,
    SUM(
```

```
        CASE
            WHEN OBC."payment_method" = 'cash' THEN
                CAN."clothing_price" + 2
            ELSE
                CAN."clothing_price"
        END
    ) AS total_price,
    OBC."payment_method"
FROM
    "Ordered By Clothing" AS OBC
LEFT JOIN
    "Clothing/Accessory item" AS CAI ON OBC."clothing_RefNb" = CAI."clothing_RefNb"
LEFT JOIN
    "Clothing/Accessory item name" AS CAN ON CAI."clothing_name" = CAN."clothing_name"

GROUP BY
    OBC."customer_email", OBC."date_of_placement", OBC."payment_method"
ORDER BY
    OBC."date_of_placement";
```

Output After Execution:



11.9. Most Ordered Items per Age Demographic:

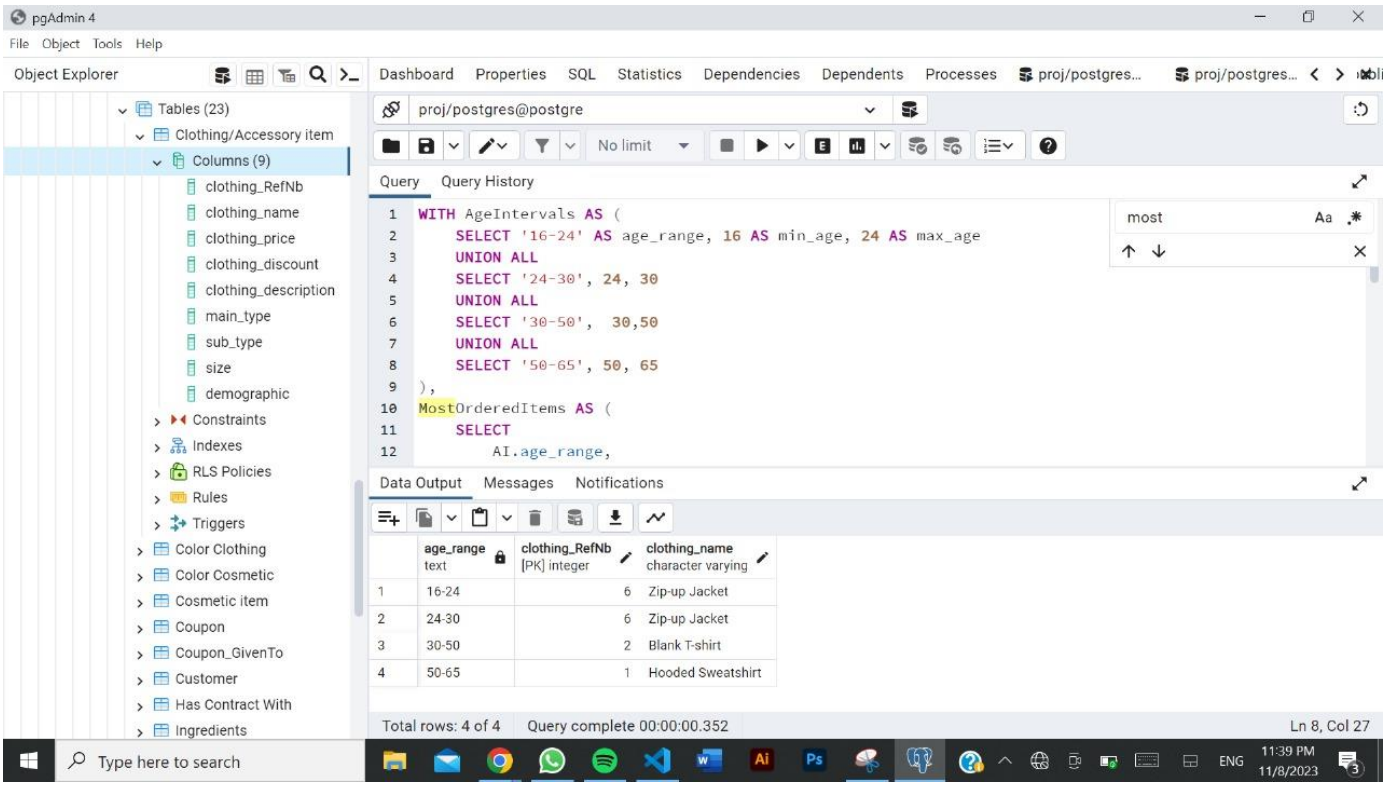
This query shows the most ordered items per relevant age demographic. This helps them know which items the customers love the most. It also helps the shop know which items to prioritize in a re-stock and in marketing campaigns.

```
WITH AgeIntervals AS (
    SELECT '16-24' AS age_range, 16 AS min_age, 24 AS max_age
    UNION ALL
    SELECT '24-30', 24, 30
    UNION ALL
    SELECT '30-50',  30,50
    UNION ALL
    SELECT '50-65', 50, 65
```

```
),
-- selecting the age intervals defining a minimum and maximum age range for each interval
MostOrderedItems AS (
    SELECT
        AI.age_range,
        CA."clothing_RefNb",
        CI."clothing_name",
        ROW_NUMBER() OVER (PARTITION BY AI.age_range ORDER BY
COUNT(OBC."clothing_RefNb") DESC) AS rn
            --Assigns a rank to each clothing item reference number within each age range based on
            --the count of orders.
    FROM
        AgeIntervals AS AI
    JOIN
        "Customer" AS C ON DATE_PART('year', AGE(C."DOB")) >= AI.min_age
        AND DATE_PART('year', AGE(C."DOB")) <= AI.max_age
-- check if customer is within age range
    JOIN
        "Ordered By Clothing" AS OBC ON C."customer_email" = OBC."customer_email"

-- match their email to the emails in the ordered by table and their ordere ref_nb to the ref_nb in the
clothing/accessory item table.
    JOIN
        "Clothing/Accessory item" AS CA ON OBC."clothing_RefNb" = CA."clothing_RefNb"
```

```
    JOIN
        "Clothing/Accessory item" AS CI ON CA."clothing_RefNb" = CI."clothing_RefNb"
    GROUP BY
        AI.age_range, CA."clothing_RefNb", CI."clothing_name"
)
SELECT
    age_range,
    "clothing_RefNb",
    "clothing_name"
--display the following for each age range in the result
FROM
    MostOrderedItems
WHERE
    rn = 1;
```

After executing:



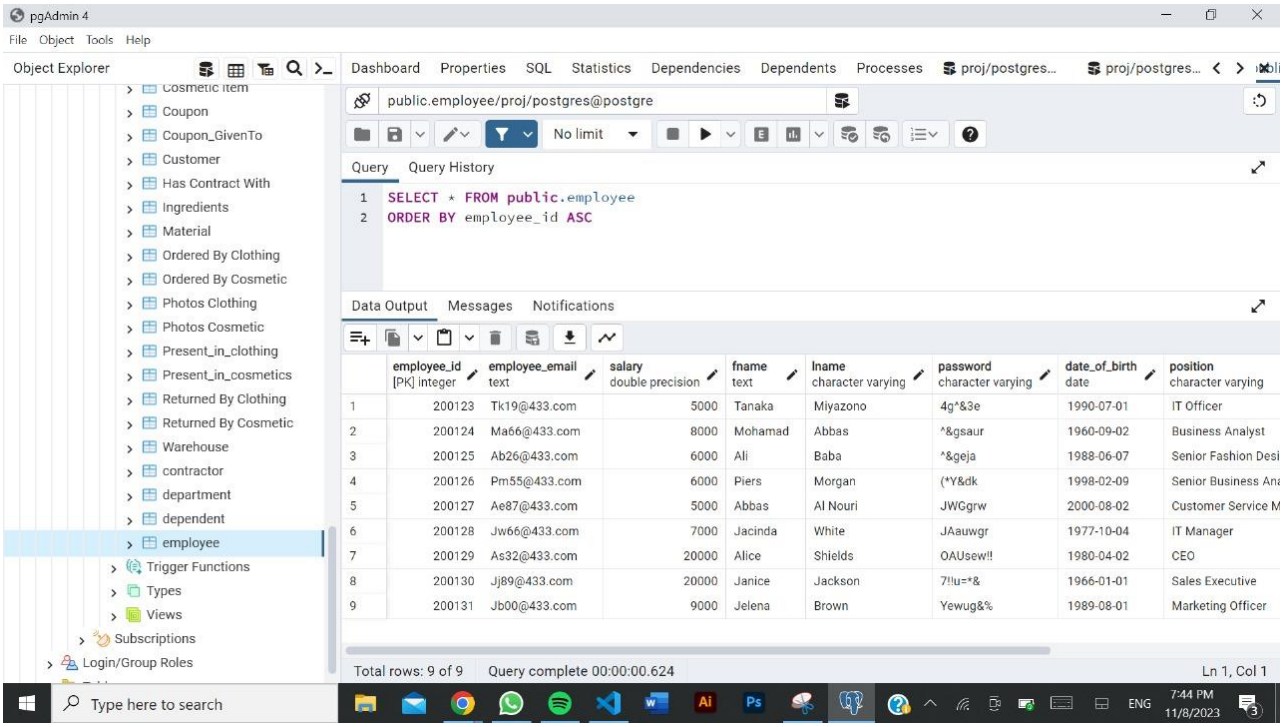
11.10. Salary Raise Depending On Dependents

This query raises the salary of an employee depending on the number of dependents they have, and each dependent's age range. For every minor dependent (between 0 and 18) it raises their salary by 1000$. For every dependent between the ages 18 and 24, the employee's salary is raised by 2000$. This compensates all employees according to their circumstances providing a healthier and overall more positive workplace.

```
-- Add $1000 for every minor dependent

UPDATE "Employee" AS E

SET salary = salary + 1000 * (

    SELECT COUNT(*)

    FROM "Dependent" AS D

    WHERE D.employee_ID = E.employee_ID

--locate employee id of dependant
```

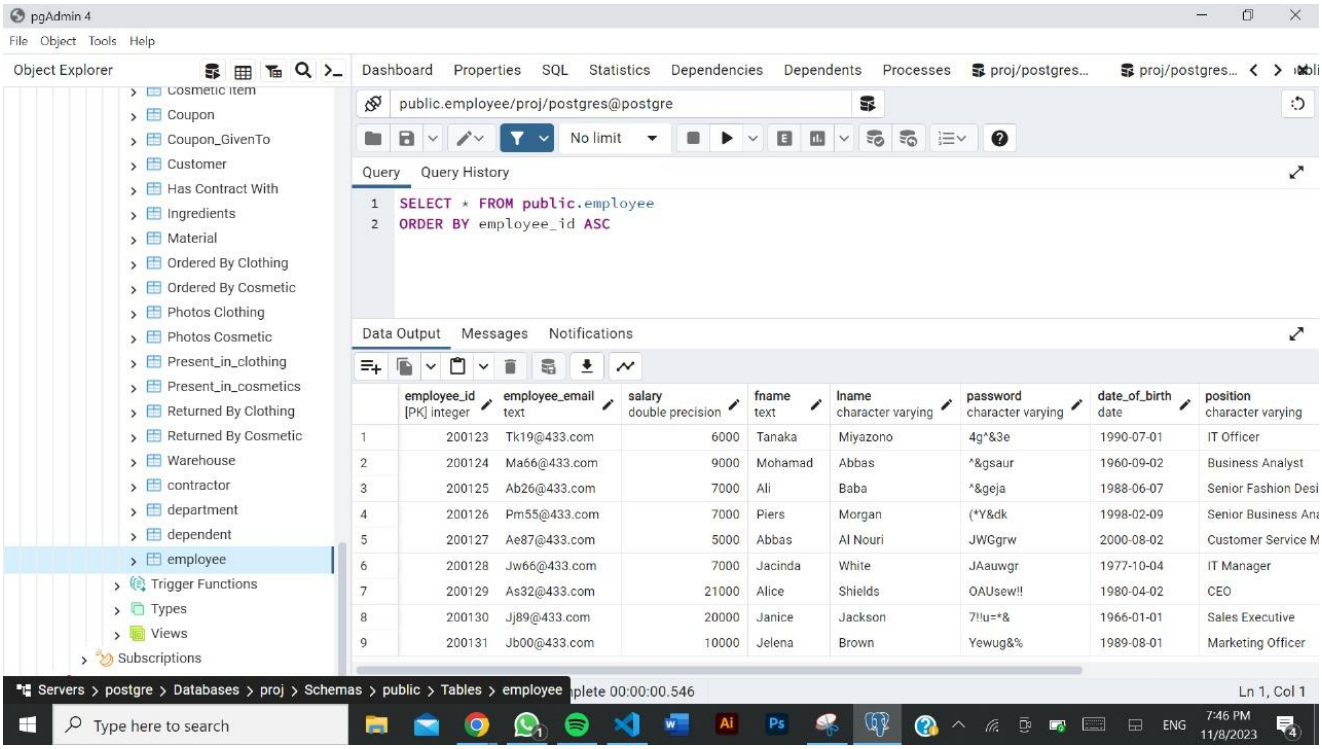
```
   AND DATE_PART('year', AGE(D.Date_of_birth)) <= 18
--check age under 18 from date of birth column in employee table
);

-- Add $2000 for every dependent between 18 and 24 who is a student
UPDATE "Employee" AS E
SET salary = salary + 2000 * (
   SELECT COUNT(*)
   FROM "Dependent" AS D
   WHERE D.employee_ID = E.employee_ID
   AND DATE_PART('year', AGE(D.Date_of_birth)) BETWEEN 18 AND 24
   );
```

Before executing:



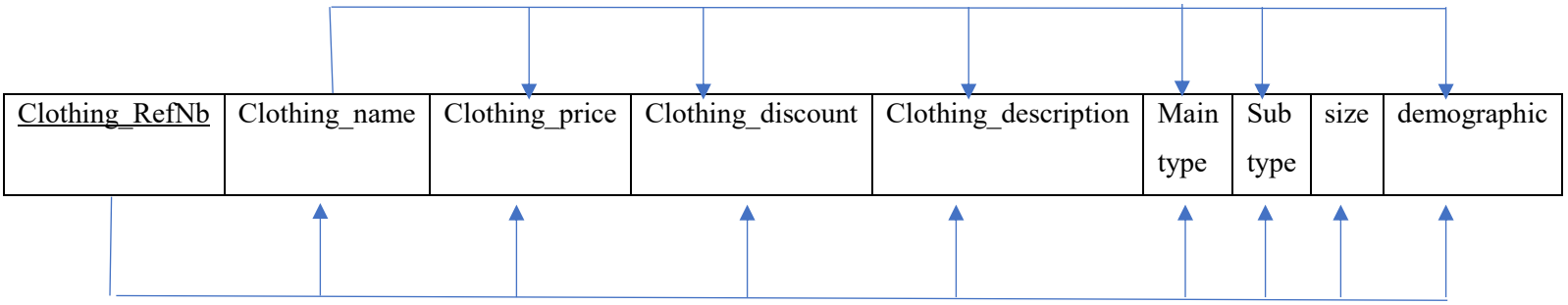
	employee_id [PK] integer	employee_email text	salary double precision	fname text	lname character varying	password character varying	date_of_birth date	position character varying
1	200123	Tk19@433.com	5000	Tanaka	Miyazono	4g^&3e	1990-07-01	IT Officer
2	200124	Ma66@433.com	8000	Mohamad	Abbas	^&gsaur	1960-09-02	Business Analyst
3	200125	Ab26@433.com	6000	Ali	Baba	^&geja	1988-06-07	Senior Fashion Desi
4	200126	Pm55@433.com	6000	Piers	Morgan	(*Y&dk	1998-02-09	Senior Business Ana
5	200127	Ae87@433.com	5000	Abbas	Al Nouri	JWGgrw	2000-08-02	Customer Service M
6	200128	Jw66@433.com	7000	Jacinda	White	JAauwgr	1977-10-04	IT Manager
7	200129	As32@433.com	20000	Alice	Shields	OAUsew!!	1980-04-02	CEO
8	200130	Jj89@433.com	20000	Janice	Jackson	7!!u=*&	1966-01-01	Sales Executive
9	200131	Jb00@433.com	9000	Jelena	Brown	Yewug&%	1989-08-01	Marketing Officer

After executing:



12. Normalization

12.1. Clothing/Accessory item



The functional dependencies within this table are represented by the above arrows. Semantically, clothing_RefNb functionally determines all attributes, as it directly and uniquely identifies an item, therefore identifying its name, price, discount, description, main and sub type, size, and its demographic. Recall that the same clothing item type can have different sizes, and each size of an item has its own

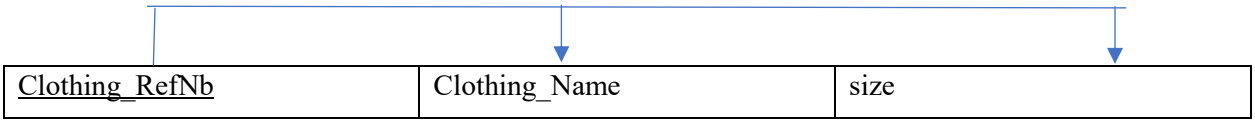
reference number. For example, assume a clothing item named “tube top” has two sizes: medium and large. The medium one has a reference number of 1234, and the large one has a reference number of 5678. Therefore, the name “tube top” is not unique, and it does not functionally determine the size nor the reference number, but it determines the price, discount, description, main and sub type, and demographic, since those are common between the medium tube top and the large tube top. This explains the functional dependencies represented by the arrows above the relation.

This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn’t contain any nested relations.

As for the 2nd normal form, it is also satisfied because the primary key contains only one attribute (clothing_RefNb) , and all attributes depend on this primary key. Therefore, all non-prime attributes fully depend on the primary key.

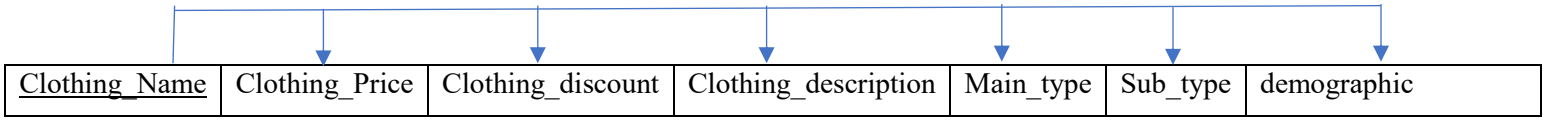
The 3rd normal form is violated, as there are transitive dependencies clothing_RefNb→Y→Z with Y being “clothing name” which is not a candidate key. (e.g. clothing_RefNb→ clothing name →clothing_description). To satisfy the 3rd normal form, the table is decomposed as follows:

Clothing/Accessory item:



Clothing_RefNb	Clothing_Name	size

Clothing/Accessory item name:

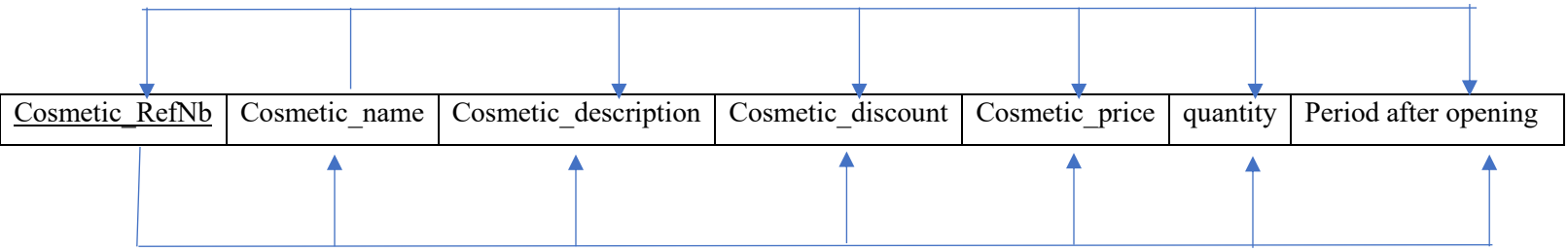


Clothing_Name	Clothing_Price	Clothing_discount	Clothing_description	Main_type	Sub_type	demographic

The newly created table “Clothing/Accessory item name” has clothing_name as its primary key. As for the old table, we removed from it the price, discount, description, main type, sub type, and demographic. Now, these tables satisfy the 3rd normal form because there are no such transitive functional dependencies.

With the newly created tables, BCNF is satisfied because for all functional dependencies X→Y, X is a super key (the super key is clothing_refNb for FDs in the first table, and clothing_name in the second).

12.2. Cosmetic item



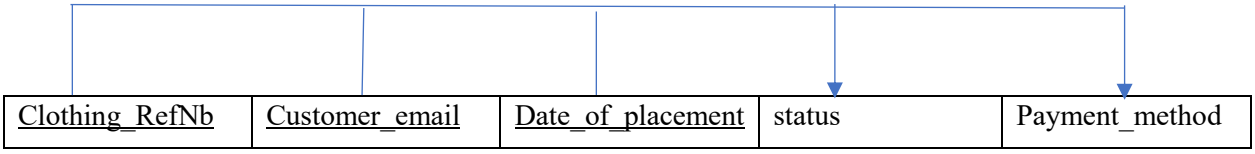
The functional dependencies are inferred as follows: cosmetic_RefNb uniquely determines a cosmetics item, thereby determining its name, description, discount, price, quantity, and period after opening. Cosmetic_name is a candidate key, as it is unique, and it also uniquely identifies an item just like "cosmetic_RefNb". Therefore, it functionally determines all other attributes.

This relation satisfies the 1st normal form as all attributes are single atomic attributes.

The 2nd Normal form is satisfied because the primary key "cosmetic_RefNb" contains only one attribute, and all non-prime attributes fully depend on it.

Notice that all FDs are of the form cosmetic_RefNb→Y or cosmetic_name→Y, and both left-hand sides are keys. Thus, it cannot be possible that a transitive functional dependency X→Y→Z exists where Y is not a key. Therefore, the 3rd normal form is satisfied. This also implies that BCNF is satisfied, as all left-hand sides are super keys.

12.3. Ordered By Clothing



The primary key (clothing_RefNb, customer_email, date_of_placement) uniquely identifies an order. Therefore, it collectively determines both status and payment method.

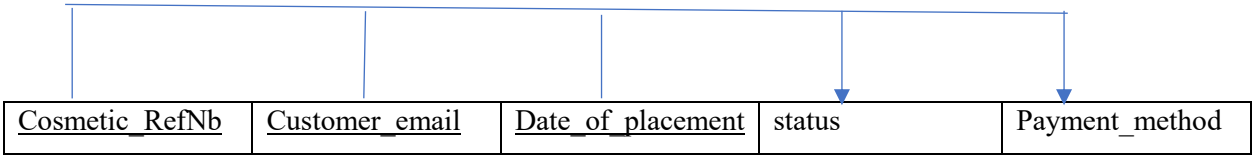
This relation satisfies the 1st normal form because all attributes are single atomic attributes.

The 2nd normal form is satisfied. "status" is fully functionally dependent on the primary key. The status of an order cannot be determined from a subset of this primary key. The same goes for payment method. The full primary key is needed to determine the payment method.

The 3rd normal form is satisfied, as there is no non-prime attribute that is transitively functionally dependent on the primary key. In fact, in all of the functional dependencies in this table, the left-hand side is always the primary key (clothing_RefNb, customer_email, date_of_placement).

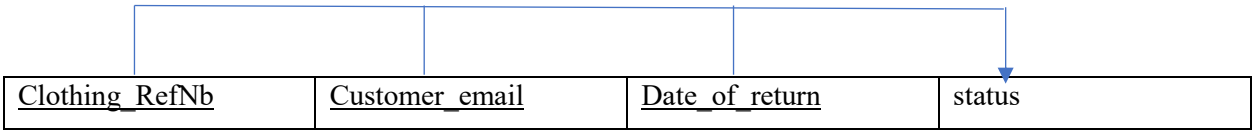
This relation satisfies BCNF because as mentioned, for all functional dependencies X→Y, X is a super key (it is always the primary key (clothing_RefNb, customer_email, date_of_placement))

12.4 Ordered By Cosmetic



This relation represents orders for cosmetics. It is exactly the same as the "Ordered By Clothing" relation, except it has a "cosmetic_RefNb" instead of "clothing_RefNb". Therefore, similarly to "Ordered By Clothing", it satisfies BCNF (see section 12.3 for more details)

12.5 Returned By Clothing



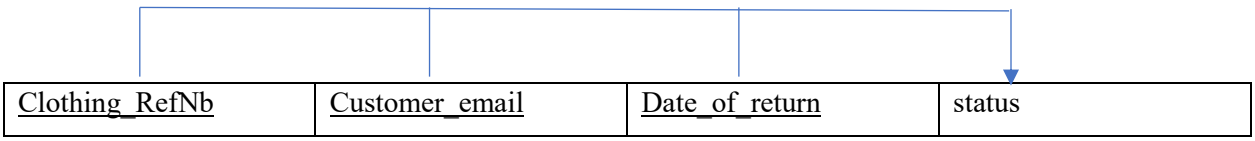
This relation satisfies the 1st normal form because there are no multivalued attributes, composite attributes, and nested relations.

It satisfies the 2nd normal form because all non-prime attributes are fully functionally dependent on the primary key (clothing_RefNb, customer_email, date_of_return). The only non-prime attribute here is status, and it cannot be determined by any subset of the primary key.

It satisfies the 3rd normal form because there is no non-prime attribute that is transitively functionally dependent on the primary key. In fact, the only FD here is (clothing_RefNb, customer_email, date_of_return) → status.

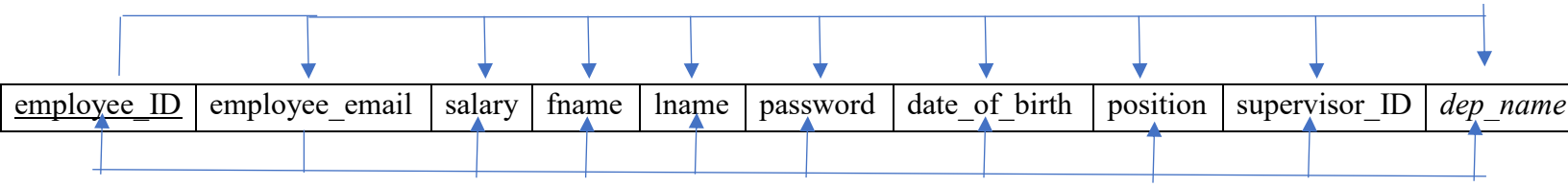
All functional dependencies (there is only one FD, as mentioned before) have super keys as their left-hand side. Therefore, this satisfies BCNF

12.6 Returned By Cosmetic



This table is equivalent to "Returned By Clothing", but it is for cosmetics. It satisfies BCNF, just like the clothing version. For more details, see section 12.5.

12.7 Employee



The functional dependencies within this table are represented by the above arrows. We see that employee_ID functionally determines all attributes, as it directly and uniquely identifies an employee. The same goes for employee_email, as it is a candidate key.

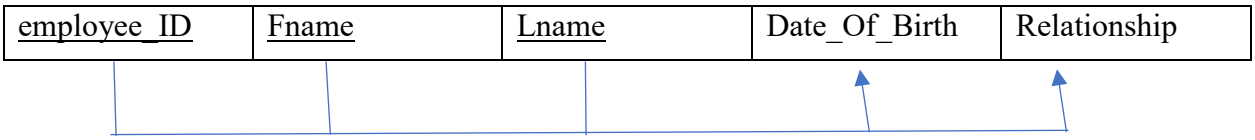
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because the primary key contains only one attribute (employee_ID) , and all attributes depend on this primary key. Therefore, all non-prime attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies involving non-prime attributes on the primary key, as all FDs are of the form employee_ID→Y or employee_email→Y.

BCNF is also satisfied as employee_ID and employee_email are indeed superkeys that uniquely identify a row in the table, and all (non-trivial) FDs are of the form employee_ID→Y or employee_email→Y. Meaning that no non-prime attribute functionally determines a prime attribute.

12.8 Dependent



The functional dependencies within this table are represented by the above arrows. It is shown that (employee_ID, Fname, Lname) functionally determines all attributes, as it directly and uniquely identifies a dependent.

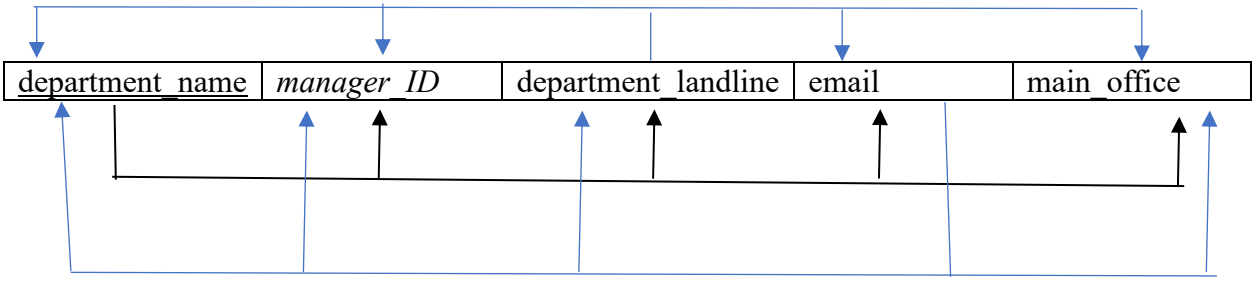
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because all attributes depend on the combination of employee id and first and last name of the dependent. From them , we can deduce the date of birth and relationship they have with the mentioned employee. Therefore, all non-prime attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies on the primary key involving non-prime attributes. Actually, non-prime attributes do not appear on the left-hand side of any FD here.

BCNF is also satisfied as (employee_ID, Fname, Lname) is indeed a superkey that uniquely identifies a row in the table, and all FDs here are of the form (employee_ID,Fname,Lname)→Y. Meaning that no non-prime attribute functionally determines a prime attribute.

12.9 Department



The functional dependencies within this table are represented by the above arrows. department_name functionally determines all attributes, as it directly and uniquely identifies a department. The same goes for email and landline, as they are candidate keys.

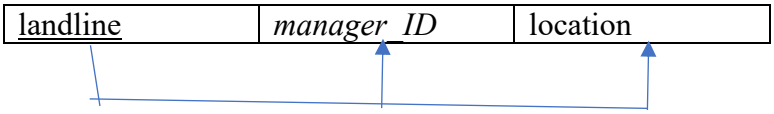
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because all attributes depend on the corresponding department name. From it, we can deduce the rest of the attributes such as the manager id, email, landline , and main office. Therefore, all non-prime attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies involving non-prime attributes on the primary key. Such transitive dependencies are not possible because all FDs have super keys on their left-hand sides.

BCNF is also satisfied as all FDs have super keys on their left-hand sides. Meaning that no non-prime attribute functionally determines a prime attribute.

12.10 Warehouse



The functional dependencies within this table are represented by the above arrows. Semantically, landline functionally determines all attributes, as it directly and uniquely identifies a warehouse.

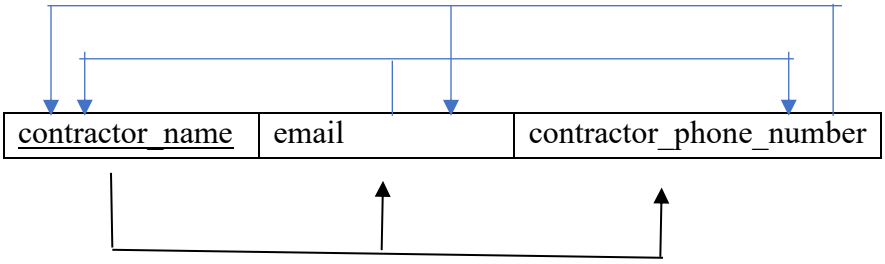
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because all attributes depend on the landline (the full primary key). From it, we can deduce the id of its corresponding manager and its location.. Therefore, all non-prime attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies involving non-prime attributes on the primary key. Such transitive dependencies cannot exist as all FDs are of the form landline→Y.

BCNF is also satisfied as landline is indeed a superkey that uniquely identifies a row in the table, and it is on the left-hand side of all FDs. Meaning that no non-prime attribute functionally determines a candidate or primary key.

12.11 Contractor



The functional dependencies within this table are represented by the above arrows. We see that contractor_name functionally determines all attributes, as it directly and uniquely identifies a dependent. The same goes for email and phone number, as they are candidate keys.

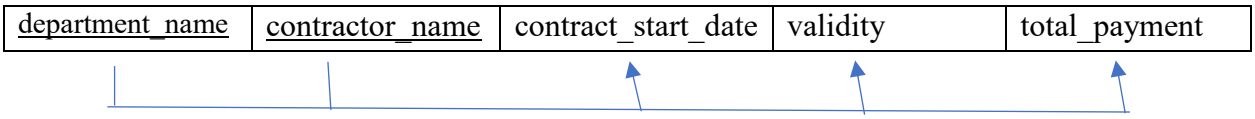
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because all attributes depend on the contractor's name (the full primary key) . From it, we can deduce their email and phone number Therefore, all attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies involving non-prime attributes on the primary key. In fact, all the attributes are keys.

BCNF is also satisfied as there are no non-prime attributes to begin with. Meaning that no non-prime attribute functionally determines a prime attribute.

12.12 Has Contract With



The functional dependencies within this table are represented by the above arrows. We deduce that (department_name, contractor_name) functionally determines all attributes, as it directly and uniquely identifies a dependent.

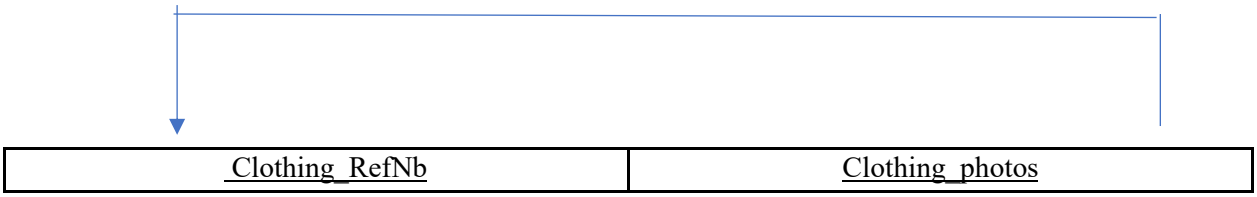
This relation satisfies the 1st normal form because it does not contain composite and multivalued attributes, and it doesn't contain any nested relations.

As for the 2nd normal form, it is also satisfied because all attributes depend on the combination of the department and the contractor's name. From them, we can deduce the contract's start date, its validity and the total payment required. Therefore, all non-prime attributes fully depend on the primary key.

As for the 3rd normal form, it is also satisfied since there are no transitive dependencies involving non-prime attributes on the primary key. Non-prime attributes do not appear on the left-hand side at all.

BCNF is also satisfied as (department_name, contractor_name) is indeed a superkey that uniquely identifies a row in the table, and all FDs are of the form (department_name, contractor_name) →Y. Meaning that no non-prime attribute functionally determines a candidate or primary key.

12.13 Photos Clothing



The primary key is (clothing_RefNb,clothing_photos), and there are no other attributes. Clothing_photos→clothing_RefNb because semantically, a photo can be of one product only.

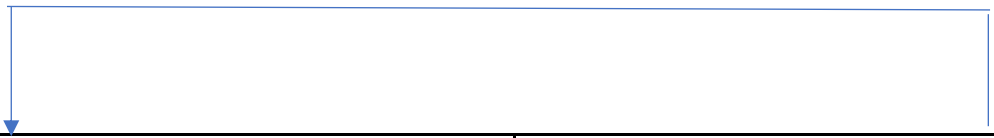
This relation satisfies the 1st normal form because all attributes are single atomic attributes.

The 2nd normal form is satisfied as there are no non-prime attributes. This means that all non-prime attributes depend fully on the primary key.

The 3rd normal form is satisfied because there is no non-prime attribute that is transitively functionally dependent on the primary key, as there are no non-prime attributes at all.

BCNF is also satisfied, for every non-trivial functional dependency X → Y in the table, X is not a non-prime attribute, as there aren't any non-prime attributes.

12.14 Photos Cosmetic



cosmetic_RefNb	cosmetic_photos

This relation represents cosmetic photos. It is exactly the same as the “Photos Clothing” relation, except it has a “cosmetic_RefNb” instead of “clothing_RefNb”. Therefore, similarly to “Photos Clothing”, it satisfies BCNF (see section 12.13 for more details)

12.15 Color Cosmetic

cosmetic_RefNb	cosmetic_color

The only FD here is trivial: (cosmetic_RefNb, cosmetic_color)→ (cosmetic_RefNb, cosmetic_color) where (cosmetic_RefNb, cosmetic_color) is the primary key

This relation satisfies the 1st normal form because all attributes are single atomic attributes.

The 2nd normal form is satisfied as all non-prime attributes are fully functionally dependent on the combined key (technically this is true, since there aren’t any non-prime attributes).

The 3rd normal form is satisfied because there is no non-prime attribute that is transitively functionally dependent on the primary key, since there aren’t any non-prime attributes.

BCNF is also satisfied, for every functional dependency X → Y in the table, X is a superkey. In this case, the composite primary key (cosmetic_RefNb,cosmetic_color) is indeed a superkey because it uniquely identifies each row in the table.

12.16 Color Clothing

clothing_RefNb	clothing_color

This relation represents clothing colors. It is exactly the same as the “Color Cosmetic” relation, except it has a “clothing_RefNb” instead of “cosmetic_RefNb”. Therefore, similarly to “Color Cosmetic”, it satisfies BCNF (see section 12.15 for more details)

12.17 Material

clothing_RefNb	material

The primary key is (clothing_RefNb,material), and the only FD here is this trivial one: primarykey→primarykey

This relation satisfies the 1st normal form because all attributes are single atomic attributes.

The 2nd normal form is satisfied as all non-prime attributes are fully functionally dependent on the combined key (technically this is true, since there aren’t any non-prime attributes).

The 3rd normal form is satisfied because there is no non-prime attribute that is transitively functionally dependent on the primary key.

BCNF is also satisfied, for every functional dependency X → Y in the table, X is a superkey. In this case, the composite primary key (clothing_RefNb,material) is indeed a superkey because it uniquely identifies each row in the table.

12.18 Ingredients

cosmetic_RefNb	ingredients

The primary key is (cosmetic_RefNb, ingredients). Similar to section 12.17, the only FD here is trivial: primarykey→primarykey.

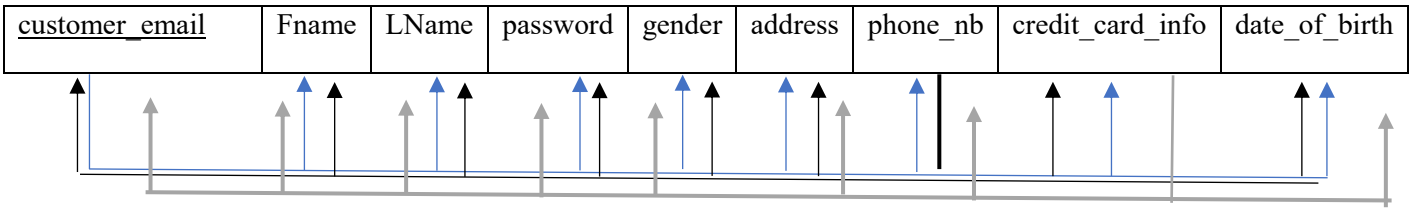
This relation satisfies the 1st normal form because all attributes are single atomic attributes.

The 2nd normal form is satisfied as all non-prime attributes are fully functionally dependent on the combined key (technically this is true, since there aren't any non-prime attributes).

The 3rd normal form is satisfied because there is no non-prime attribute that is transitively functionally dependent on the primary key.

BCNF is also satisfied, for every non-trivial functional dependency X → Y in the table, X must be a super key. In this case, the composite primary key (cosmetic_RefNb, ingredients) is indeed a super key because it uniquely identifies each row in the table.

12.19 Customer



In this relationship, customer_email, phone_nb, and credit_card_info are all keys and they uniquely identify all the information about the customer. The only primary key is the customer's email and the candidate keys that are phone number and credit card information do determine everything as well.

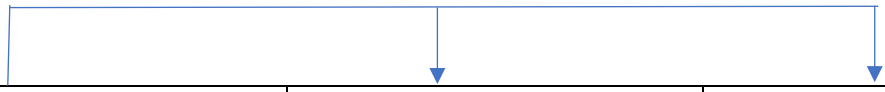
This relation satisfies the 1st normal form because all its attributes are atomic(it does not contain composite and multivalued attributes as well as any nested relations).

The 2nd normal form is also satisfied because the primary key contains only one attribute (customer_email) , and all attributes depend on this primary key. Therefore, all non-prime attributes and secondary keys fully depend on the primary key.

The 3rd normal form is satisfied as there aren't any transitive dependencies(no non-prime attribute is transitively dependent on the primary key), since the only FDs are the ones where key attributes determine all other attributes.

This relationship also satisfies BCNF since for any functional dependency X → Y, X is a superkey and in this case, X is either customer_email, phone_nb, or credit_card_info which form superkeys.

12.20 Coupon



code	type	nb_of_times_used

In this relationship, the coupon's code, the primary key, uniquely identifies all the information about the coupon(its type and the number of times this coupon has been used).

This relation satisfies the 1st normal form because all its attributes are atomic(it does not contain composite and multivalued attributes as well as any nested relations). The only primary key is the code and there aren't any secondary keys.

The 2nd normal form is also satisfied because the primary key contains only one attribute that is the code , and all attributes depend on this primary key. Therefore, all non-prime attributes and secondary keys fully depend on the primary key.

The 3rd normal form is satisfied as there aren't any transitive dependencies(no non-prime attribute is transitively dependent on the primary key).

This relationship also satisfies BCNF since for any functional dependency $X \rightarrow Y$, X is a superkey and in this case, X is always code which is the primary key.

12.21 Coupon Given To

code	customer_email

This relationship is composed of a primary key that is a super key (code, customer_email) and there aren't any non-trivial dependencies.

Then, this relation satisfies the 1st normal form because all attributes are single atomic attributes.

As for the 2nd normal form, it is satisfied since both "code" and "customer" form the primary key and there are no other attributes that are functionally dependent on those keys. There are no non-prime attributes. Thus, we can say that all non-prime attributes fully depend on the primary key.

The 3rd normal form is satisfied because there isn't any non-prime attribute that is transitively functionally dependent on the primary key.

BCNF is also satisfied, for every non-trivial functional dependency X → Y in the table, X must be a superkey. In this case, the composite primary key is (code, customer_email) and the table is only composed of the superkey without any other attribute.

12.22 Present in Clothing

clothing_RefNb	landline	quantity

This relationship is composed of a primary key that is (clothing_RefNb,landline) and it collectively determine quantity.

Then, this relation satisfies the 1st normal form because all attributes are single atomic attributes.

As for the 2nd normal form, it is satisfied since both "clothing_RefNb" and "landline" form the primary key and "quantity" is fully functionally dependent on both clothing_RefNb and landline.

The 3rd normal form is satisfied because there isn't any non-prime attribute that is transitively functionally dependent on the primary key.

BCNF is also satisfied, since for every non-trivial functional dependency X → Y in the table, X is a superkey. In this case, it is always the composite primary key is (clothing_RefNb,landline), and it determines all other attributes(in this case "quantity"). This is the only non-trivial FD.

12.23 Present in Cosmetic

cosmetic_RefNb	landline	quantity

This relation is the same as the "Present in Clothing" relation, except it has a "cosmetic_RefNb" instead of "clothing_RefNb". Therefore, similarly to "Present in Clothing", it satisfies BCNF (see section 12.22)

13. Conclusion

To summarize, the entities and relationships discussed give a clear view of the database that is to be developed. After creating the design and delving into the specifics and importance of every item that was added to form a cohesive ER diagram for the company FOUR3THREE, and after developing the mapping of the ER and implementing the database on PostgreSQL and normalizing the relations, the database application is to be developed next.