AMERICAN UNIVERSITY OF BEIRUT

MAROUN SEMAAN FACULTY OF ENGINEERING & ARCHITECTURE

American University of Beirut

School of Engineering and Architecture

Department of Electrical and Computer Engineering

A Database Design for NexStore Company

By

Jad Shaker (Group Leader)

jss31@mail.aub.edu

Hamza Atout

hsa60@mail.aub.edu

Khaled Ammoura

kaa74@mail.aub.edu

A REPORT

submitted to Dr. Hussein Bakri in partial fulfillment of the requirements of phase 2 of the database project

for the course **EECE433 – Database Systems**

October 2024

Contents

List of Figures

List of Tables

List of Code Snippets

1 Introduction

”Data is the new oil, and databases are the engines that refine it”. In today’s competitive retail world, firms must manage both their physical and online operations efficiently to fulfill customer expectations and maximize resources. The design of a full database is important to ease the integration of inventory, sales, and customer information, and provide connection operations between both channels. This report portrays the architecture of a database for a retail business that operates in both physical and online modes, represented through an Entity-Relationship (ER) diagram.

The main goal of this database is to offer flawless coordination across the store’s physical inventory and its online site. The ER diagram highlights the connection between significant entities such as Customers, Products, Orders, and Employees demonstrating how data interacts within the system. It acts as the core of the database that maintains efficient retail processes, ensuring correctness, consistency, and scalability.

This report represents NexStore company’s main database design using an Entity-Relationship diagram. The References section contains all the used citations. The Tools section describes all the instruments used to draw database design. The System Description and Requirements section defines all the needs of the client. The Legend of ER diagram symbols include all the symbols of the used notation. The complete ER diagram is presented in the ER diagram for the NexStore database section. In the subsection ”Entity Types & Their Attributes”, we elucidate the various types and attributes of each entity. In the other subsection ”Relationships and Their Explanations”, all relationships are listed and explained. In the ”Conclusion” section, we summarize the main points of the design document.

2 References / Copyright Section

References

[1] *Elmasri, R., & Navathe, S.* Fundamentals of Database Systems, 7th edition.

[2] *Dr. Hussein Bakri.* EECE433 - Database Systems Slides.

[3] *draw.io.* Draw.io `https://draw.io/`

[4] *LaTeX.* LaTeX `https://latex-project.org/`

[5] *Logo.com.* LogoAI `https://logo.com/`

[6] *OpenAI.* ChatGPT `https://chatgpt.com/`

[7] *GitHub Copilot.* GitHub `https://github.com/features/copilot/`

[8] *Claude AI.* Claude AI `https://claude.ai/`

3 Tool Used to draw the ER Diagram

- The ER diagram was drawn using the online tool draw.io. [3]

- The report was written using LaTeX. [4]
- The logo was created using LogoAI. [5]

4 System Description & Requirements

1. Supplier is identified by supplier's name, contact information including email, name and phone number, and website.
2. Product is identified by SKU, price, name, description, and image URLs that might include several images, colors, weights, brands, and dimensions (width, height, length).
3. The category is identified by name and description.
4. The employee is identified by SSN, hire date, date of birth, gender, address, phone number, email, name, position, and salary.
5. Address consists of country state, street, building, and apartment.
6. The branch is identified by phone number, name, address, and work hours (opening hours, closing hours) of the branch which might have different values depending on the day.
7. The customer is identified by phone number, date of birth, address, hashed password to ensure security, date of registration, email, name, and gender.
8. Order is identified by order ID, notes, payment method, total amount, and whether the order is online.
9. The department is identified by name, locations (it might have several locations), and an updated number of employees.
10. Driver is identified by license number, driving experience years, and license expiry date.
11. The coupon is identified by code, discount percentage, number of times used, minimum and maximum order amount, usage limit, description, and time interval of validation (valid to, valid from).
12. Suppliers could supply zero or more products, and every product should be supplied by exactly one supplier.
13. Every product should be listed under exactly one category, which might be a subcategory of exactly one parent category.
14. An order must contain at least one product, and a product could be contained in several orders.
15. The relationship between the product and the order takes the quantity and the amount -in USD- of the product as attributes.
16. Every order is made by exactly one customer; however, a customer could have several orders.
17. The relationship between the customer and the order takes the date of when the order was processed.

18. Every product must be in at least one branch, but a branch could have many products.
19. The relationship between the products and branches takes the quantity of the product and on which shelf it is placed as attributes.
20. A customer might review many products, and a product could be reviewed by several customers.
21. Reviews relationship takes the review date, rating, and image URLs that might have several images, comments, and descriptions as attributes.
22. Every employee should work in exactly one branch, and a branch should have one or many employees.
23. Every employee should be supervised by exactly one other employee, and an employee could supervise many employees.
24. Every branch should be managed by exactly one employee, and an employee might manage a branch.
25. An order must be physically checked out by exactly one employee, and an employee could physically check out many orders.
26. Every department should have at least one employee, and each employee should work for exactly one department.
27. All relationships between the departments and the employees include the date when the employee started working for the department as an attribute.
28. Every department should be managed by exactly one employee, but an employee could manage a department.
29. An employee might have some dependents, but a dependent must depend on exactly one employee.
30. Dependent is identified by name, gender, date of birth, and his/her relationship to the employee.
31. A driver is an employee, but not every employee is a driver.
32. Every order should be delivered by exactly one driver, but a driver could deliver several orders.
33. The relationship between the driver and the order takes the address, and actual and expected time of delivery as attributes.
34. An order could be redeemed by one coupon, and a coupon could be redeemed by one order.
35. The relationship between the order and the coupon takes the redemption date and discount amount -in USD- as attributes.
36. The relationship wishlist must be requested by a customer and could be empty or could include several products.
37. Wishlist takes as an attribute the total amount of the products.

38. The relationship between the customer and the support ticket takes the requested date as an attribute.

39. The support ticket is identified by ticket number, description, subject, status, and priority.

40. Every support ticket should be assigned to exactly one employee, but an employee could be assigned to many support tickets.

41. The relationship between the employee and the support ticket takes the assignment date as an attribute.

5 Legend of ER Diagram Symbols

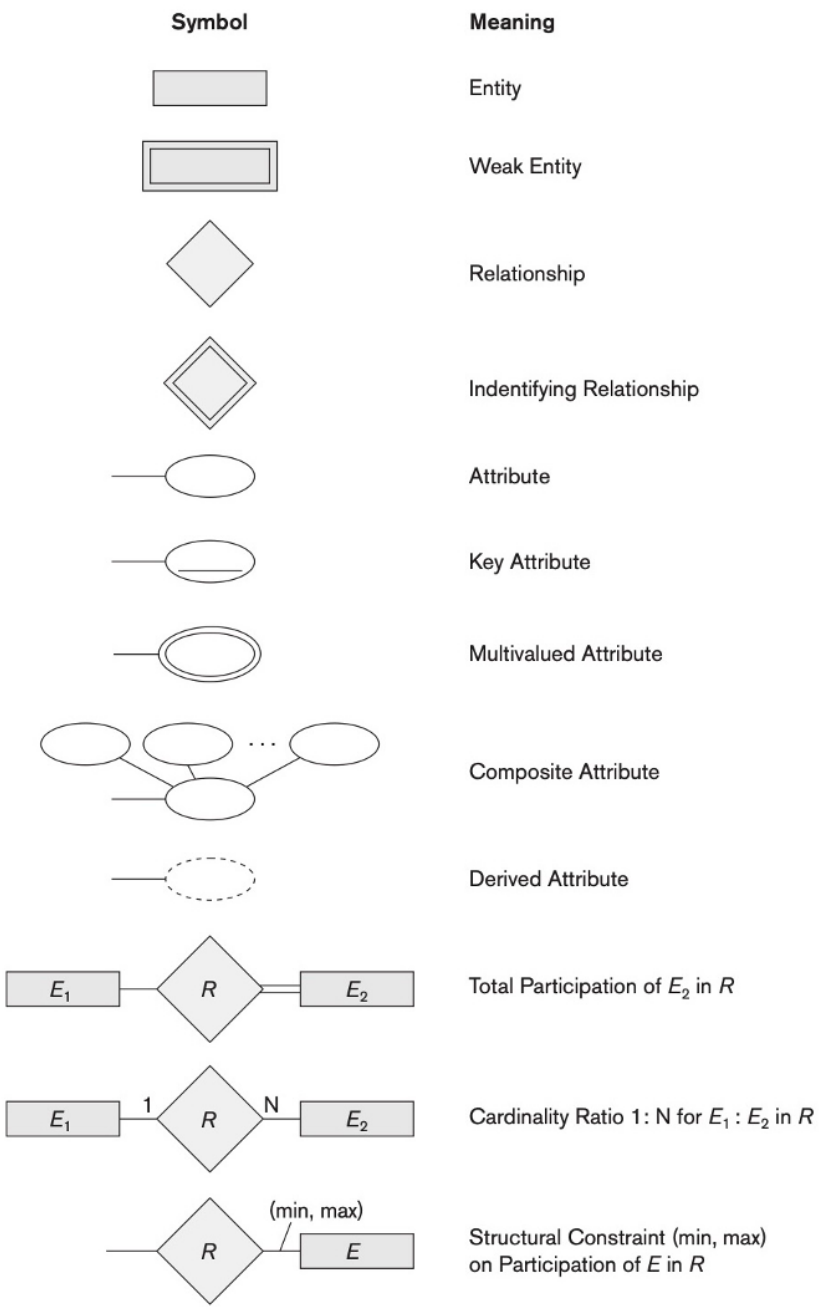


Figure 5.1: *Legend of ER Diagram Symbols*

6 ER Diagram for the NexStore Database

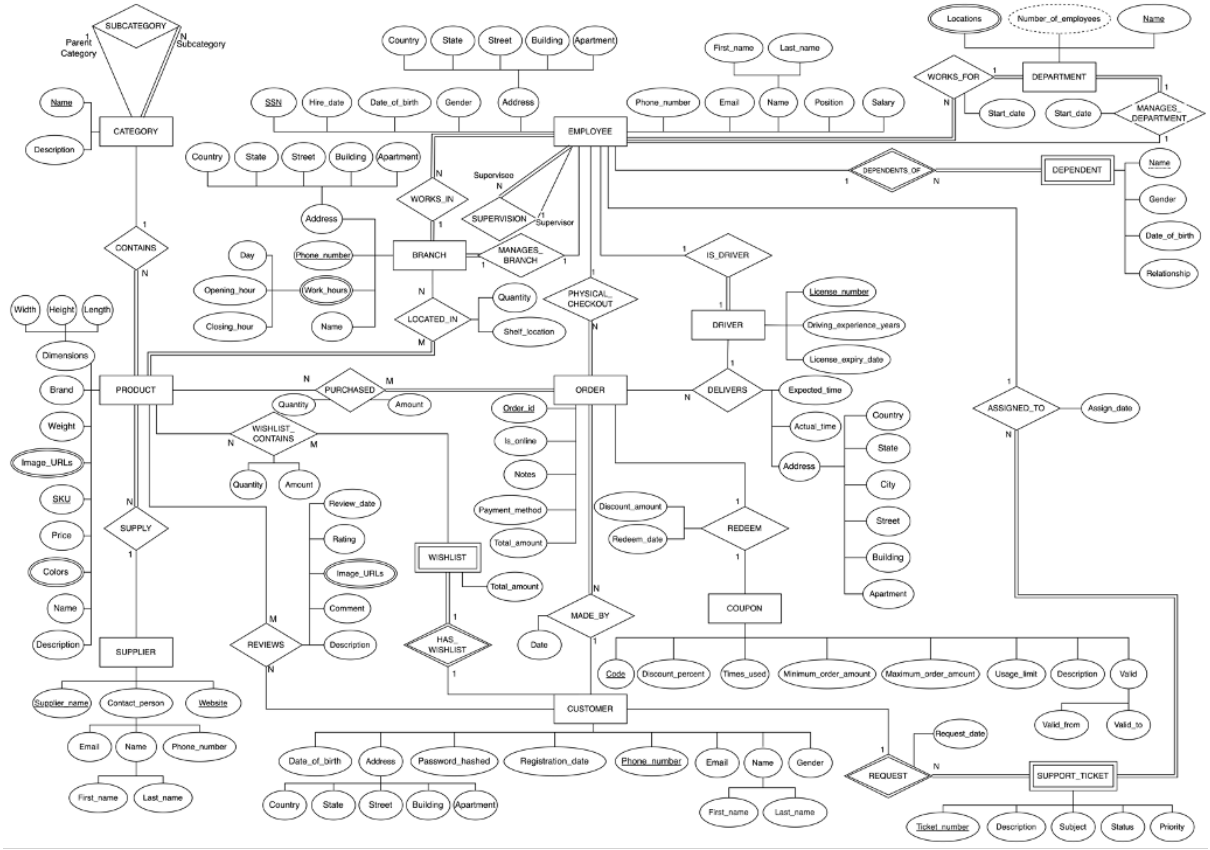


Figure 6.1: *ER Diagram for the NexStore Database*

7 New Complete Amended ER Diagram for the NexStore Database

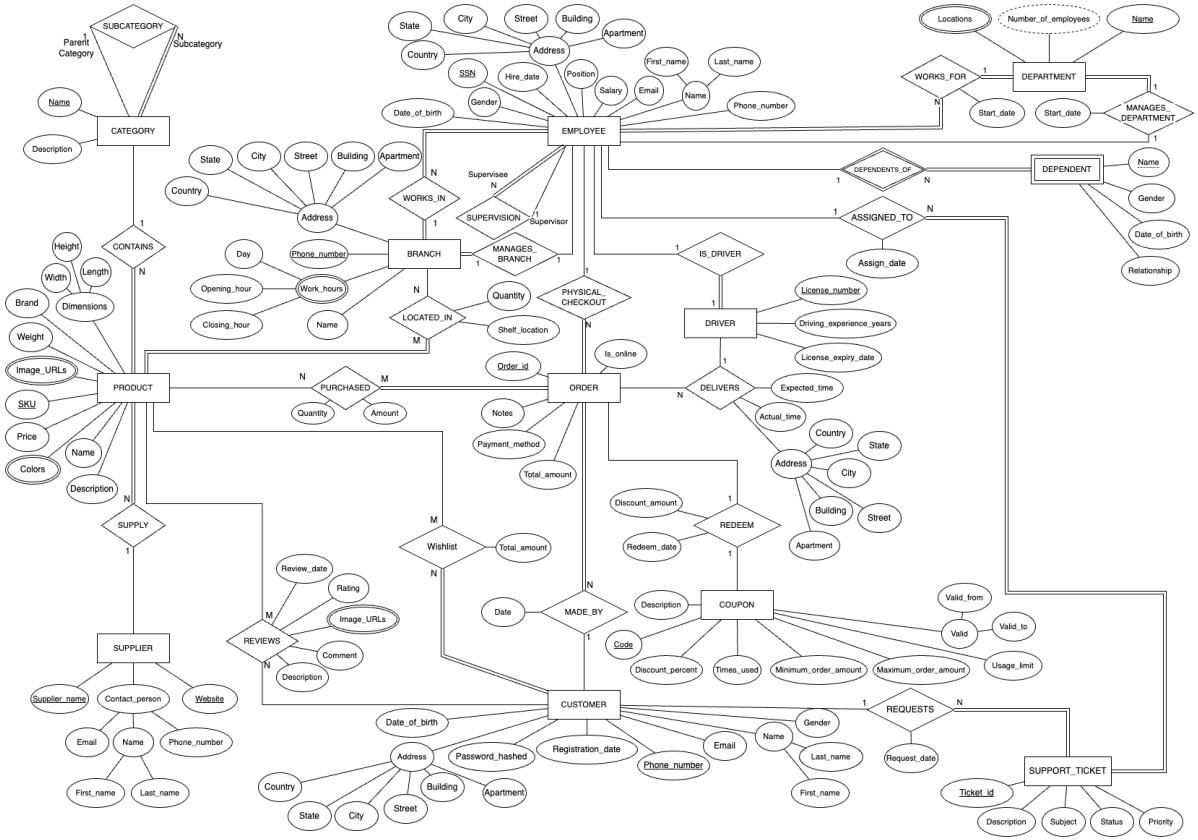


Figure 7.1: *ER Diagram for the NexStore Database*

7.1 Entity Types and Their Attributes

7.1.1 Branch

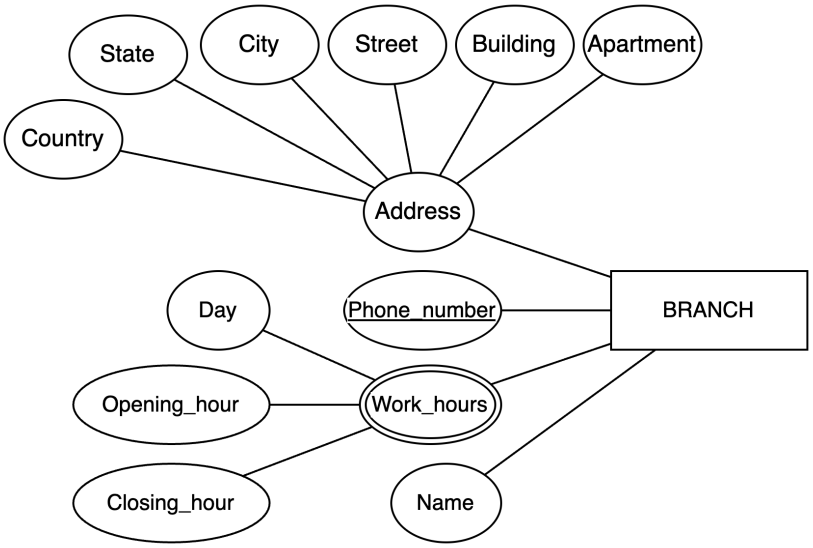


Figure 7.2: *Branch Entity and its Attributes*

The branch is one of the main entities presented in the ER diagram. The phone number was the primary key chosen to uniquely identify each branch. Additionally, the work hours were selected to be a composite multi-valued attribute since each branch might have different work hours depending on the weekday (e.g., weekends have different work hours). It comprises the weekday and the opening and closing hours of the branch. We included the composite attribute address to indicate the accurate address of the branch. Finally, we added the name attribute that indicates the name of each branch.

7.1.2 Category

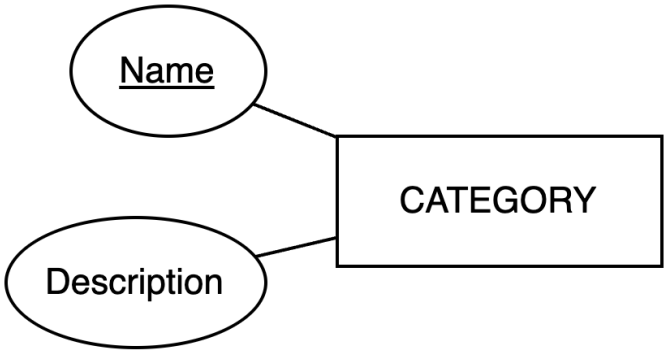


Figure 7.3: *Category Entity and its Attributes*

As it is essential to know the category of each product, we include the category entity. It contains the Name of the category as a primary key in addition to the description of the category.

7.1.3 Coupon

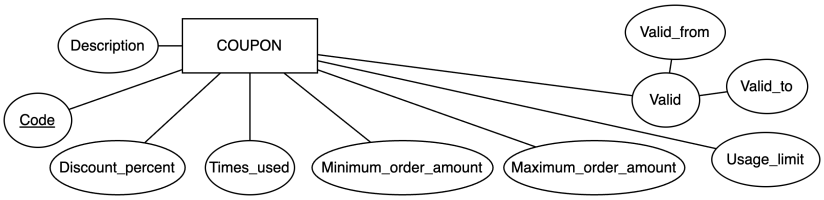


Figure 7.4: *Coupon Entity and its Attributes*

In several events, a coupon might be applied to the order. The coupon can be identified by its unique code, so we chose the code to be the key attribute. Each coupon can be applied a certain number of times on a specific amount ranging from a minimum to a maximum, so we added the four attributes: ”Times used”, ”Usage limit”,” Minimum order amount” and ”Maximum order amount”. Furthermore, since each coupon is valid for a specific time interval, we included the ”Valid to” and ”Valid from” attributes. Finally, we added the attributes ”Discount percent” and ”Description” to indicate the discount percentage and the description of each coupon.

7.1.4 Customer

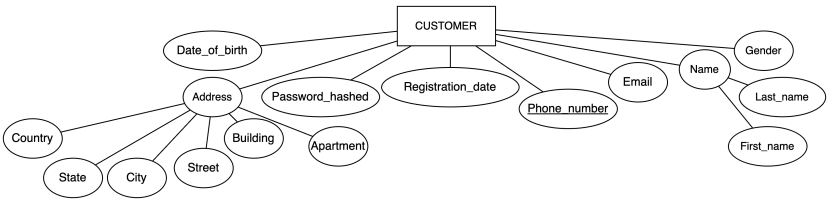


Figure 7.5: *Customer Entity and its Attributes*

This entity describes all the information we need to know about the customer comprising several attributes. We chose the phone number of the customer to be the primary key since it uniquely identifies each customer. Additionally, we included the name of the customer as a composite attribute as it contains the first and last name. Similarly, we added the address attribute that consists of the county, state, city, street, building, and apartment of the customer. Moreover, we added the password hashed to ensure security. Furthermore, we added the email attribute to ensure communication between the customer and the store. Finally, we added the gender, registration date, and date of birth of the customer that can be used for special events such as the customer’s birthday.

7.1.5 Department

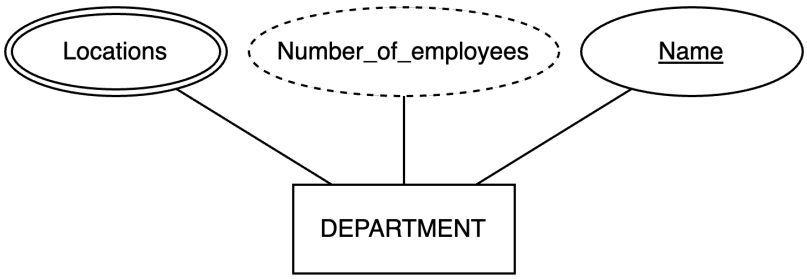


Figure 7.6: *Department Entity and its Attributes*

This entity represents the departments in the company. Each department might have several locations, so this attribute was multi-valued. Since each department has a unique name, we chose the name to be the key attribute. Finally, we added the derived attribute ”number of employees” to keep track of the updated number of employees in each department.

7.1.6 Dependent

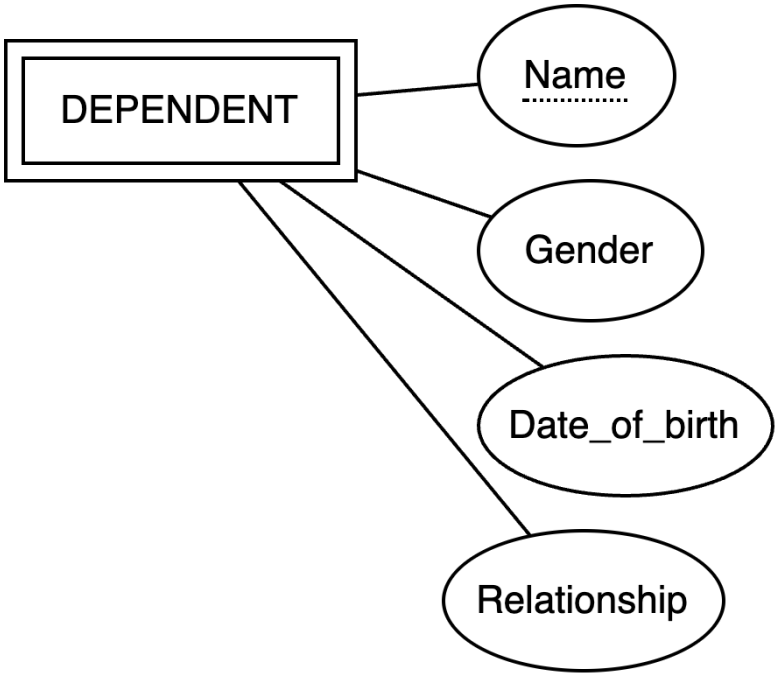


Figure 7.7: *Dependent Entity and its Attributes*

Each employee has dependents that are related to them. Since we cannot have a dependent without having an employee, we set the dependent entity to be weak with the name attribute as a weak attribute of it.

7.1.7 Driver

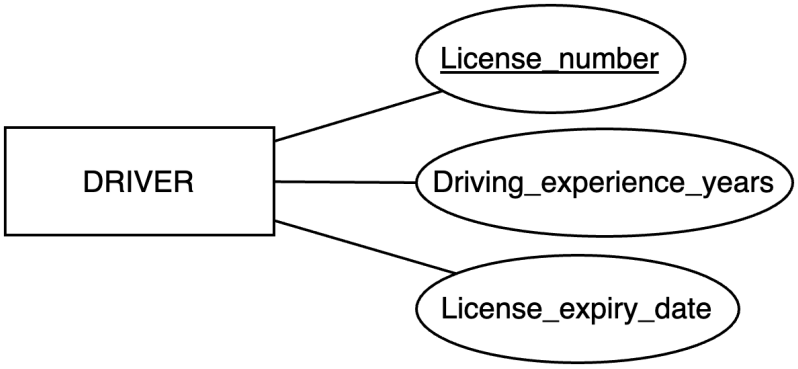


Figure 7.8: *Driver Entity and its Attributes*

To deliver an online order, a driver needs to be assigned. This driver entity has the license number as a primary key since it uniquely identifies the driver. Additionally, other attributes reflecting information about the driver are the driving experience years and the license expiry date.

7.1.8 Employee

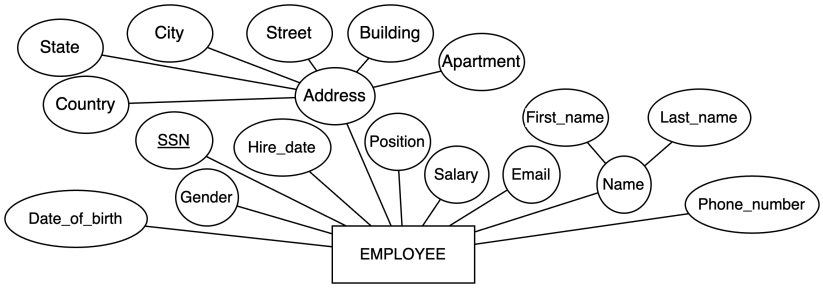


Figure 7.9: *Employee Entity and its Attributes*

One of the basic entities in this project is the employee entity which portrays all the information needed about the employee. We chose the social security number (SSN) of the employee as the primary key since it uniquely identifies each employee. Additionally, we added the address attribute that consists of the county, state, city, street, building, and apartment of the employee. Moreover, we included the name of the customer as a composite attribute as it contains the first and last name. Finally, we added all the information needed such as date of birth, phone number, position, gender, email address, salary, and hire date to keep an eye on this important personal information.

7.1.9 Order

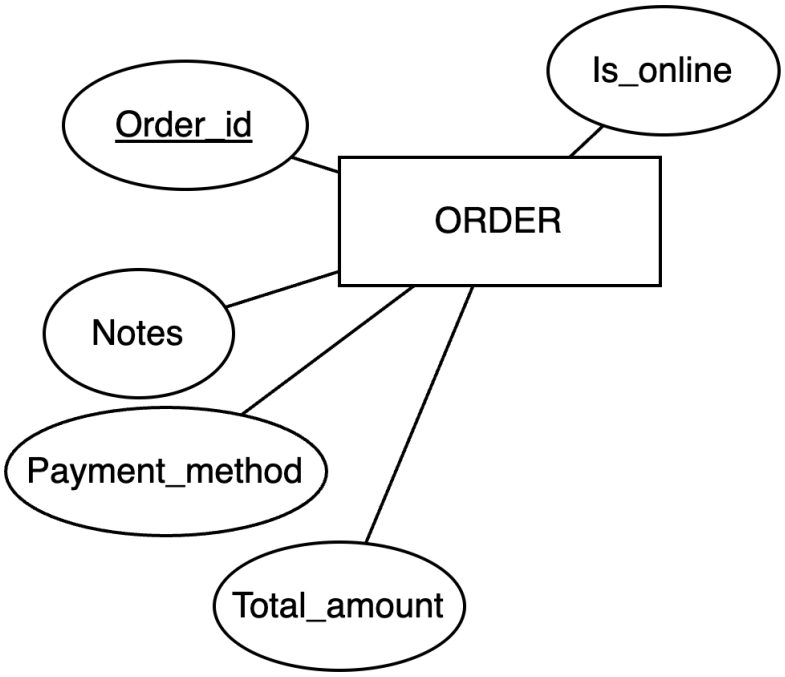


Figure 7.10: *Order Entity and its Attributes*

To proceed with the customers' orders on both physical and online channels, the order entity was added. The primary key of this entity is the unique order ID. Other attributes include the total cost of the order, the payment method used, and any notes for this order. Finally, we added the "is online" attribute to specify whether the order was conducted physically or online.

7.1.10 Product

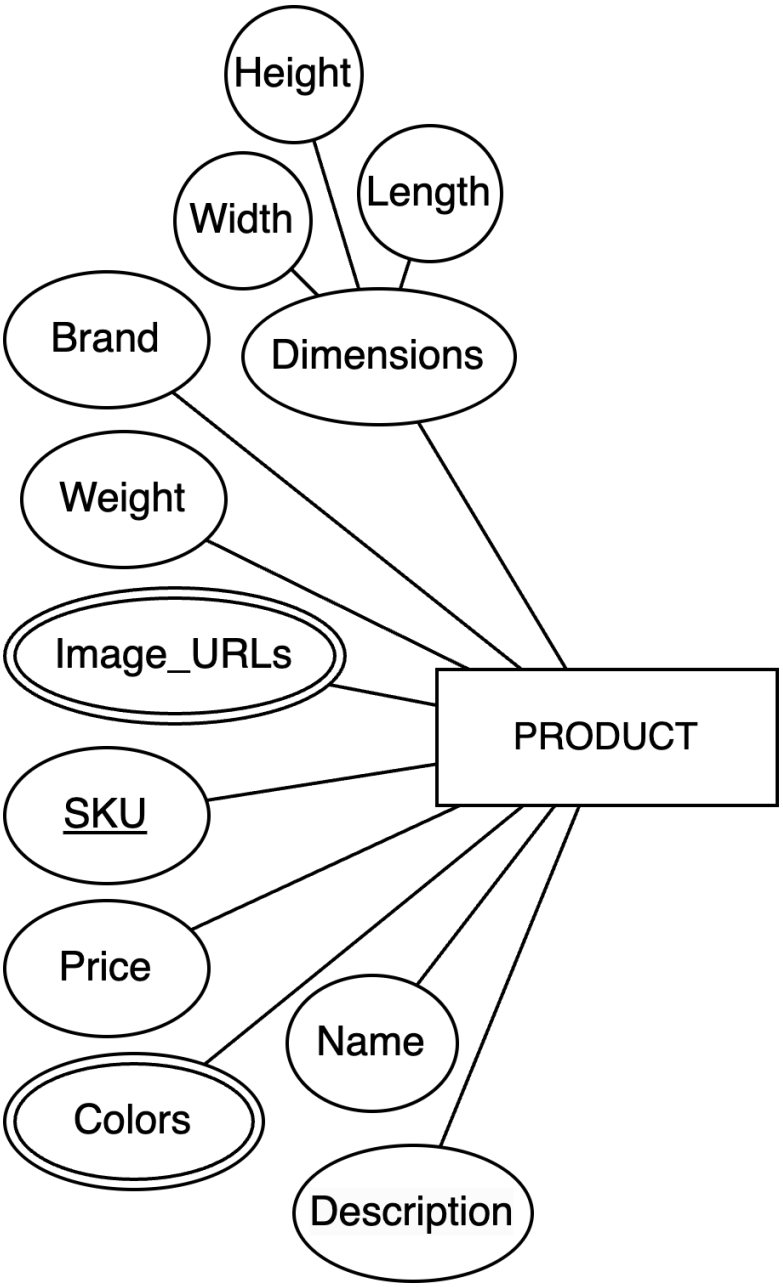


Figure 7.11: *Product Entity and its Attributes*

This product entity describes all the information we need to know about the product comprising several attributes. We set the product entity as a weak entity since it doesn't exist without the dependence on the branch entity. We chose the stock-keeping unit (SKU) of the product as the primary key since it uniquely identifies each product. Additionally, we included the dimensions of the product as a composite attribute as it contains the height, width, and length of the product. Moreover, we added the colors as a multi-valued attribute since one product could have various colors. Similarly, we added the image URLs as a multi-valued attribute since a product could have several images to cast it online. Furthermore, we

added the weight to keep track of the total weight of the shipment. In addition, we used the quantity to view the available amount of this product. Finally, we added the name, description, price, brand, and the date when the product was added to specify these essential parameters of each product.

7.1.11 Supplier

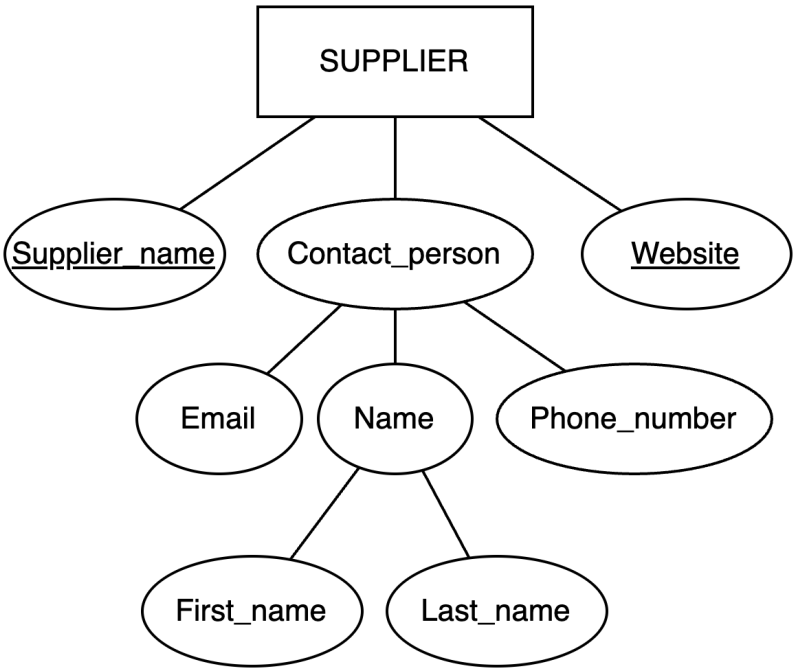


Figure 7.12: *Supplier Entity and its Attributes*

Another basic entity in this project is the supplier entity which shows all the information needed about the employee. We chose the supplier's website and name as the primary keys since they uniquely identify each supplier. Finally, we included the contacted person as a composite attribute as it contains a composite attribute, the name composing the first and last name, email, and supplier's phone number.

7.1.12 Support Ticket

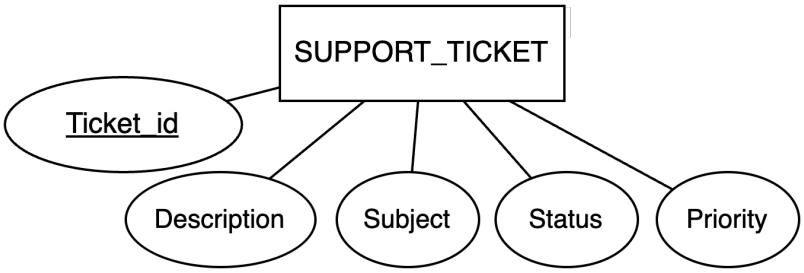


Figure 7.13: *Support Ticket Entity and its Attributes*

To keep up with the customers' complaints, a support ticket entity was needed. We used the ticket ID as a primary key as it uniquely identifies the support ticket. Moreover, other attributes like subject, description, priority -to know how urgent the request is-, and status to keep track of it were needed.

7.2 Relationships and their Explanations

7.2.1 Assigned To

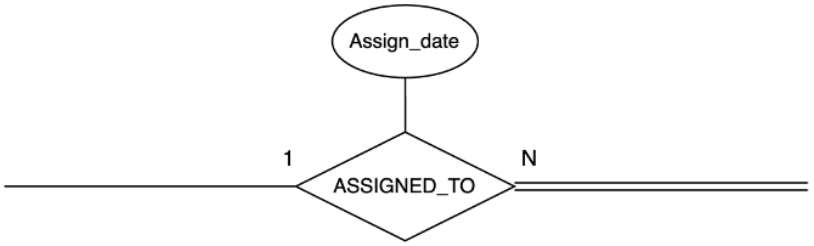


Figure 7.14: *Assigned To Relationship*

The relationship "assigned to" is between the employee and the support ticket. It maps the employee to zero or more support tickets and stores the data of the assignment. Also, it supports the real-life need that a support ticket must be assigned to an employee.

7.2.2 Contains

Figure 7.15: *Contains Relationship*

The relationship "contains" is between a category and products. A category may contain many products. This organized the products the company has by categorizing all the products under specific categories. Also, it helps a better user experience by searching the category and then looking for the specific product needed in the category.

7.2.3 Delivers

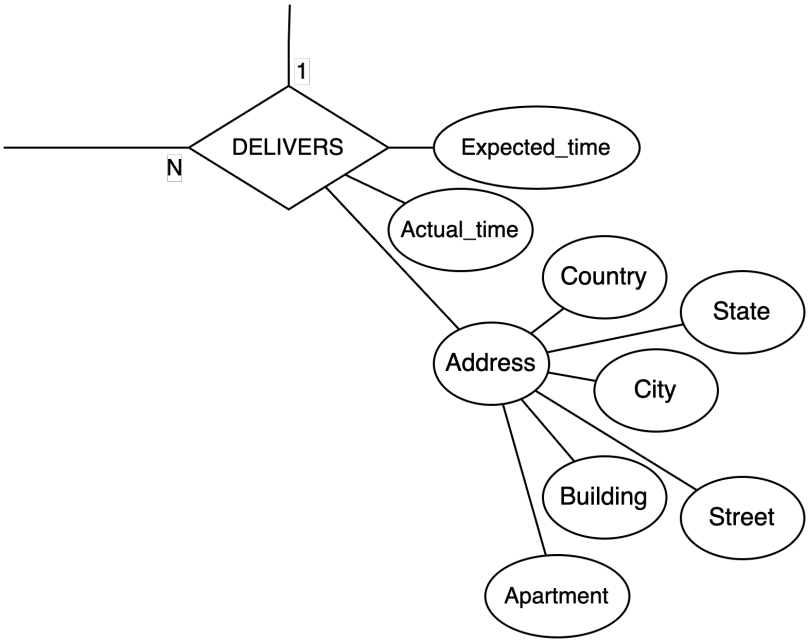


Figure 7.16: *Delivers Relationship*

The relationship ”Delivers” is between the Driver and the Order. The driver is responsible for delivering the order to the requested address provided by the customer. It includes the ”expected time” attribute to be displayed to the customer and the ”actual time” attribute to help predict accurate delivery times for future orders. A driver can deliver many orders, but each order is delivered by one driver. Some orders may be physically checked out, in which case no delivery is required.

7.2.4 Dependents Of

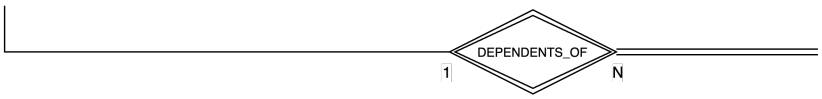


Figure 7.17: *Dependents Of Relationship*

The ”Dependents Of” relationship is between the Employee and the Dependent. Each employee may have several dependents. This relationship is essential for connecting employees to their dependents, who might be insured by the company in the future.

7.2.5 Is Driver

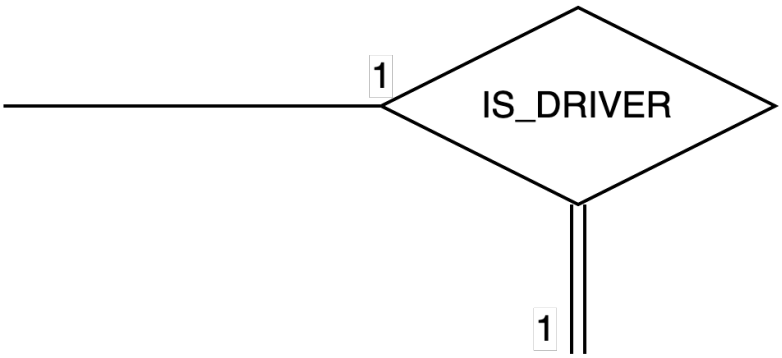


Figure 7.18: *Is Driver Relationship*

The ”Is Driver” relationship is between the Employee and the Driver. Each driver is also an employee of the company, and this relationship helps categorize employees based on whether they are drivers.

7.2.6 Located In

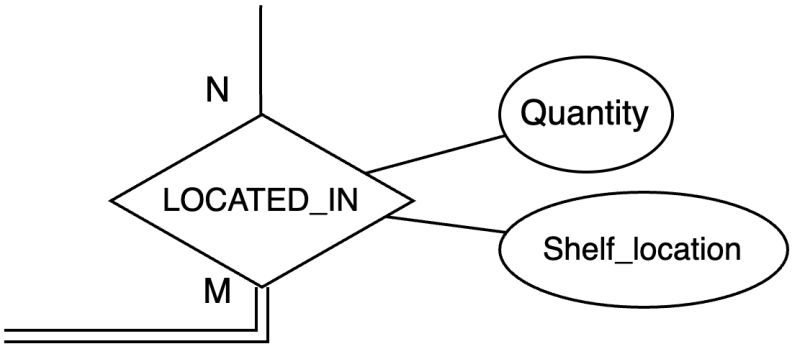


Figure 7.19: *Located In Relationship*

The ”Located In” relationship is between the Product and the Branch. A branch can stock multiple products, each with specific quantities placed on certain shelves. The quantity and shelf location may vary between branches to avoid redundancy.

7.2.7 Made By

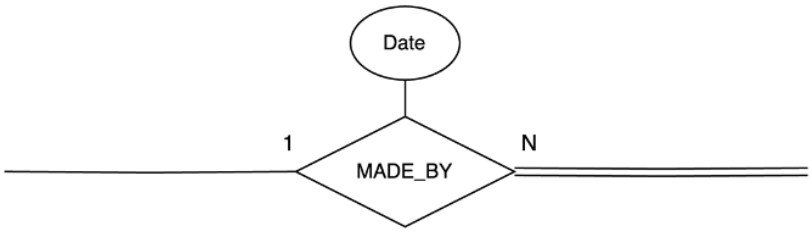


Figure 7.20: *Made By Relationship*

The ”Made By” relationship connects the Customer and the Order. Each order is placed by one customer, but a customer can place multiple orders. This relationship also stores the order date for tracking purposes.

7.2.8 Manages Branch

Figure 7.21: *Manages Branch Relationship*

The ”Manages Branch” relationship is between the Employee and the Branch. Each branch is managed by one employee, but not all employees are managers. This relationship helps define the company’s management structure.

7.2.9 Manages Department

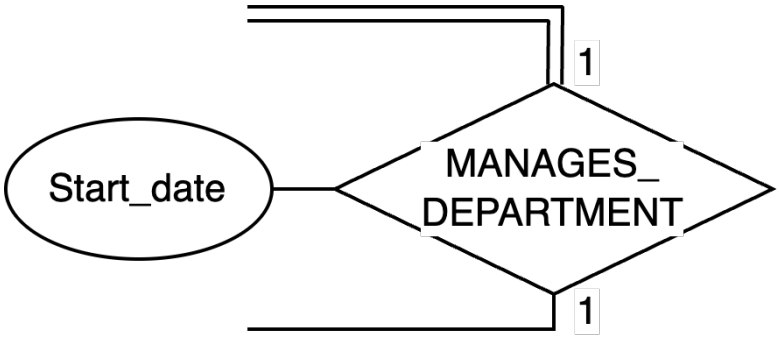


Figure 7.22: *Manages Department Relationship*

The ”Manages Department” relationship is between the Employee and the Department. An employee can manage one department, enabling a clear tracking of department managers.

7.2.10 Physical Checkout

Figure 7.23: *Physical Checkout Relationship*

The ”Physical Checkout” relationship is between the Employee and the Order. Each order must be checked out by one employee, but an employee can check out multiple orders. This relationship prevents duplicate receipts and streamlines order management.

7.2.11 Purchased

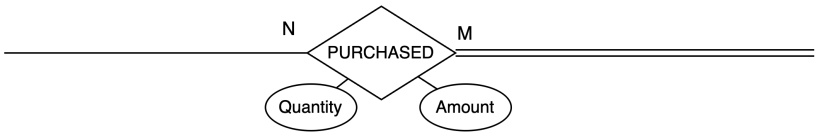


Figure 7.24: *Purchased Relationship*

The ”Purchased” relationship is between the Product and the Order. An order can contain multiple products, each with specific quantities and prices. This relationship tracks these details to calculate the total order value.

7.2.12 Redeem

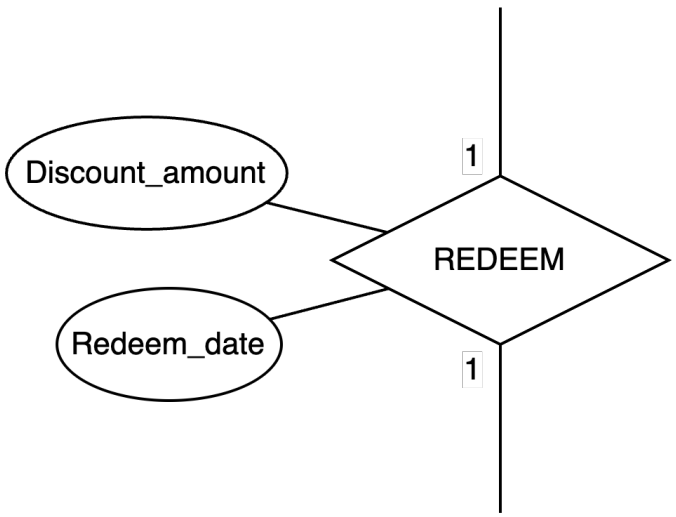


Figure 7.25: *Redeem Relationship*

The ”Redeem” relationship connects the Order and the Coupon. An order can be redeemed using one coupon, and a coupon can only redeem one order. This relationship supports special occasion discounts.

7.2.13 Request

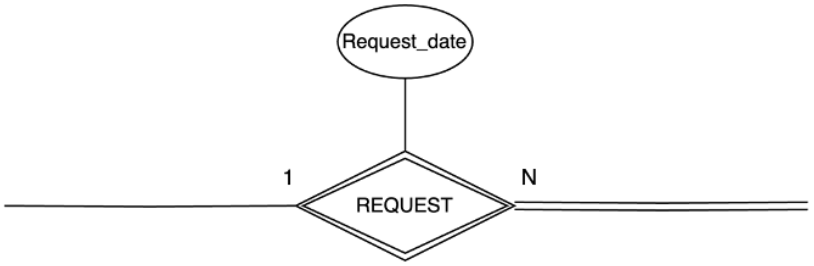


Figure 7.26: *Request Relationship*

This relationship is between the customer and the support ticket. It enables the customer to create various support tickets, but a ticket can’t be created without the request of the customer. By including this relationship in our system, it manages the complaints of the customers efficiently.

7.2.14 Reviews

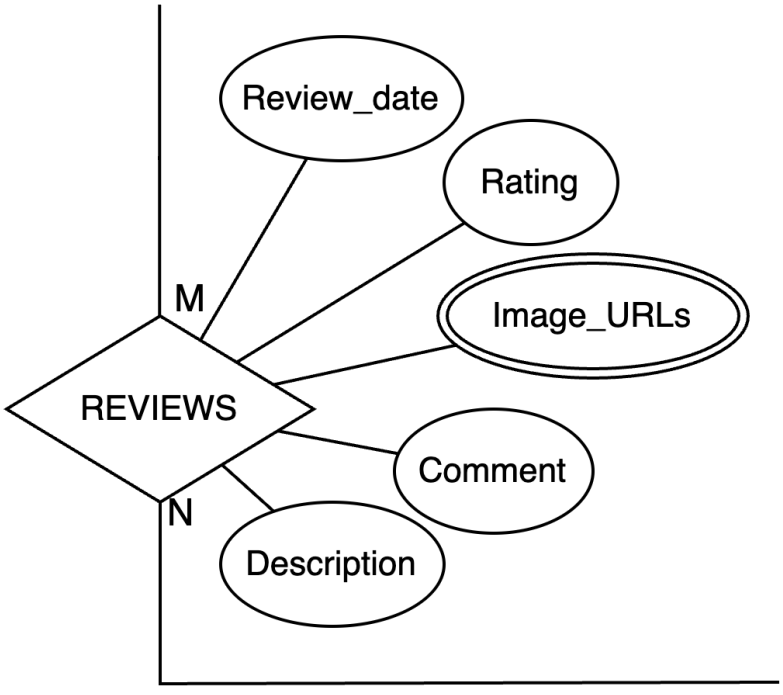


Figure 7.27: *Reviews Relationship*

The ”Reviews” relationship connects the Customer and the Product. A customer can review several products, and a product can receive reviews from multiple customers. This helps gather feedback for product improvement.

7.2.15 Subcategory

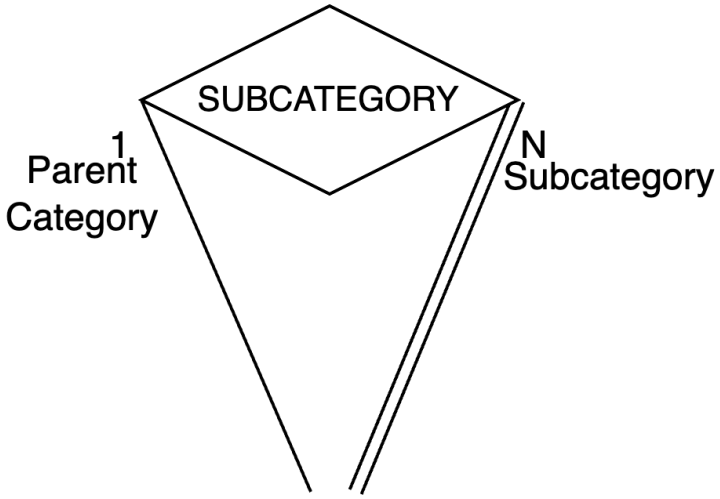


Figure 7.28: *Subcategory Relationship*

The ”Subcategory” relationship connects a Category to its subcategories. This relationship ensures hierarchical organization of product categories.

7.2.16 Supply

Figure 7.29: *Supply Relationship*

The ”Supply” relationship is between the Supplier and the Product. A supplier can supply multiple products. It tracks the quantity, date, and price of supplied products.

7.2.17 Supervision

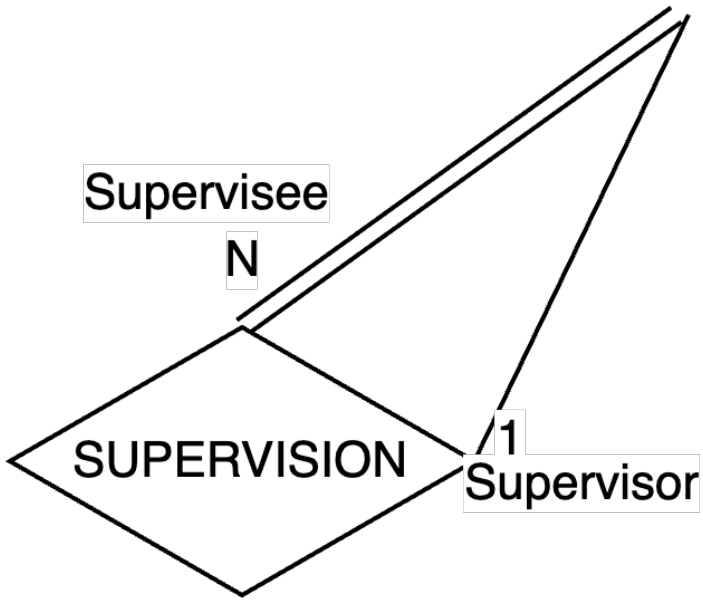


Figure 7.30: *Supervision Relationship*

The ”Supervision” relationship is a recursive relationship among Employees. An employee can supervise others, ensuring a clear management hierarchy.

7.2.18 Wishlist

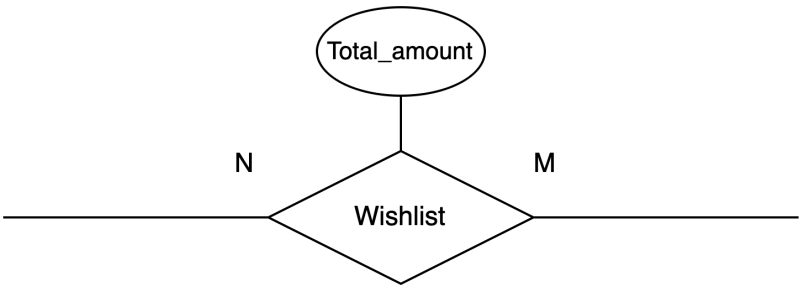


Figure 7.31: *Wishlist Relationship*

The ”Wishlist” relationship is between the customer and the product. A wishlist must be created by a customer. However, a wishlist can be empty, or it can include several products. It takes as an attribute the total amount (price) of the products. This relationship ensures that the customer can save the items he/she wishes to obtain later on.

7.2.19 Works For

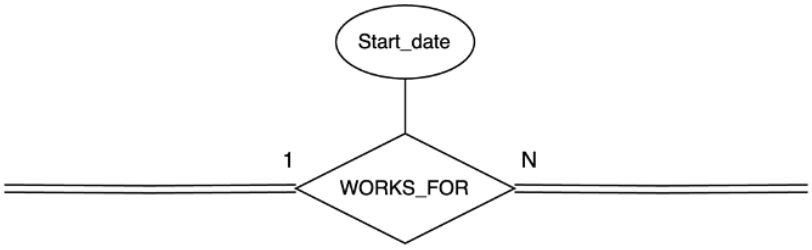


Figure 7.32: *Works For Relationship*

The ”Works For” relationship connects the Employee and the Department. Many employees work for one department, and each department must have at least one employee.

7.2.20 Works In

Figure 7.33: *Works In Relationship*

The ”Works In” relationship links the Employee and the Branch. A branch must have at least one employee, and each employee works in one branch.

8 ER to Relational Mapping

8.1 Mapping of Strong Entity Types

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple (atomic) attributes of E.
- Choose one of the key attributes of E as the primary key for R.
- If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of the relation R. [2]

8.1.1 Branch

Branch (Phone_number, Name, Country, State, City, Street, Building, Apartment)

We created the relation Branch including various attributes of the strong entity Branch. We included all the simple (atomic) attributes of Branch that are Name and Phone_number. Moreover, we decomposed the composite attribute Address into simple attributes as country, state, city, street, building , and apartment. Furthermore, we didn’t include the composite multi-valued attribute which is Work_hours as

we'll create its own relation later. Finally, we chose the key attribute Phone_number to be the primary key that uniquely identifies the relation Branch.

8.1.2 Category

Category (Name, Description)

We created the relation Category including the two atomic attributes of the strong entity Category that are Name and Description. Additionally, we chose the key attribute Name as the primary key.

8.1.3 Coupon

Coupon (Code, Description, Discount_percent, Times_used, Minimum_order_amount, Maximum_order_amount, Usage_limit, Valid_from, Valid_to)

We created the relation Coupon including attributes of the strong entity Coupon. We included all atomic attributes which are Code, Description, Discount_percent, Times_used, Minimum_order_amount, Maximum_order_amount, and Usage_limit. Furthermore, we decomposed the composite attribute Valid into two simple attributes which are Valid_from and Valid_to. Finally, we chose the key attribute Code as a primary key.

8.1.4 Customer

Customer (Phone_number, Email, First_name, Last_name, Gender, Registration_date, Password_hashed, Date_of_birth, Country, State, City, Street, Building, Apartment)

We created the relation Customer including different attributes of the strong entity Customer. We included all the atomic attribute which are Phone_number, Email, Gender, Registration_date, Password_hashed and Date_of_birth. Moreover, we decomposed the two composite attribute Name and Address as First_name and Last_name and Country, State, City, Street, Building and Apartment, respectively.

8.1.5 Department

Department (Name, Number_of_employees)

We created the relation Department including various attributes of the strong entity Department. We included all the simple attributes which are Name and Number_of_employees, such that Name acts a primary key and Number_of_employees as a derived attribute. Furthermore, we didn't include the multi-valued attribute which is Locations as we'll create its own relation later.

8.1.6 Driver

Driver (License_number, Driving_experience_years, License_expiry_date)

We created the relation Driver including all the atomic attributes of the strong entity Driver which are: the key attribute which is License_number as a primary key, Driving_experience_years and License_expiry_date.

8.1.7 Employee

Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email, First_name, Last_name, Phone_number, Country, State, City, Street, Building, Apartment)

We created the relation Employee including various attributes of the regular entity Employee. We included all the simple attributes which are: SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email and Phone_number. Moreover, we decomposed the two composite attribute Name and Address as First_name and Last_name and Country, State, City, Street, Building and Apartment, respectively. Finally, we chose the key attribute SSN as the primary key.

8.1.8 Order

Order (Order_id, Notes, Payment_method, Total_amount, Is_online)

We created the relation Order including all the atomic attributes of the strong entity Order which are: the key attribute which is Order_id as a primary key, Notes, Payment_method, Total_amount and Is_online.

8.1.9 Product

Product (SKU, Name, Price, Description, Weight, Brand, Width, Height, Length)

We created the relation Product including various attributes of the regular entity Product. We included all the simple attributes which are SKU, Name, Price, Description, Weight and Brand. Moreover, we decomposed the composite attribute Dimensions into three atomic attributes which are: Width, Height and Length. Furthermore, we didn't include the multi-valued attributes which are Image_URLs and Colors as we'll create relations from them later. Finally, we chose the key attribute SKU as the primary key.

8.1.10 Supplier

Supplier (Website, Supplier_name, Contact_person_email,
Contact_person_first_name, Contact_person_last_name, Contact_person_phone_number)

We created the relation Supplier including various attributes of the regular entity Supplier. We included all the simple attributes which are: Supplier_name and Website. Moreover, we decomposed the composite attribute into two atomic attributes and a composite attribute, which are Email and Phone_number and Name, respectively. Furthermore, we decomposed Name into two atomic attributes which are First_name and Last_name. Finally, we chose the key attribute Website as the primary key.

8.1.11 Support Ticket

Support_Ticket (Ticket_id, Description, Subject, Status, Priority)

We created the relation Support_Ticket including attributes of the regular entity Support_Ticket. We included all the simple attributes which are: Ticket_id, Description, Subject, Status and Priority. Finally, we chose the key attribute Ticket_id as the primary key.

8.2 Mapping of Weak Entity Types

- According to Elmasri and Navathe, in Fundamentals of Database Systems (2015), "For each weak entity type W in the ER schema with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any." [1]

8.2.1 Dependent

Dependent (*Employee_SSN*, Name, Gender, Date_of_birth, Relationship)

We created a relation Dependent for the weak entity type Dependent with employee entity type Employee. We included all the atomic attributes of Dependent which are Name, Gender, Date_of_birth, and Relationship. Moreover, we created the foreign key Employee_SSN which references to the primary key of Employee which is SSN. Finally, we assigned the tuple Employee_SSN and the weak attribute Name as the primary key of this relation.

8.3 Mapping of Binary 1:1 Relationship Types

- According to Elmasri and Navathe, in Fundamentals of Database Systems (2015), "For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- Foreign Key approach: Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S." [1]

8.3.1 Redeem

Coupon (Code, Description, Discount_percent, Times_used, Minimum_order_amount,
Maximum_order_amount, Usage_limit, Valid_from, Valid_to, *Order_ID*,
Discount_amount_ Redeem_date)

The 1:1 relationship Redeem is mapped by choosing the participating entity type Coupon to serve in the role of S, because both participating entity types have a partial participation in the Redeem relationship type, so, it doesn't matter which one we choose. Moreover, we chose Order_ID as the foreign key referencing to the primary key of Order. Finally, we added all the atomic attributes of the relationship Redeem to the relation Coupon.

8.3.2 Manages Branch

Branch (Phone_number, Name, Country, State, City, Street, Building, Apartment, *Employee_SSN*)

The 1:1 relationship Manages_branch is mapped by choosing the participating entity type Branch to serve in the role of S, because its participation in the Manages_branch relationship type is total. Moreover, we added the foreign key Employee_SSN referencing to the primary key of Employee.

8.3.3 Is Driver

Driver (License_number, Driving_experience_years, License_expiry_date, *Employee_SSN*)

The 1:1 relationship Is_driver is mapped by choosing the participating entity type Driver to serve in the role of S, because its participation in the Is_driver relationship type is total. Moreover, we add the foreign key Employee_SSN referencing to the primary key of Employee.

8.3.4 Manages Department

Department(Name, Number_of_employees, *Employee_SSN*, Manager_start_date)

The 1:1 relationship Manages_department is mapped by choosing the participating entity type Department to serve in the role of S, because its participation in the Manages_department relationship type is total. Moreover, we add the foreign key Employee_SSN referencing to the primary key of Employee. Finally, we added the atomic attribute of the relationship Manages_department to the relation Coupon.

8.4 Mapping of Binary 1:N Relationship Types

- According to Elmasri and Navathe, in Fundamentals of Database Systems (2015), "For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- Include any simple attributes of the 1:N relation type as attributes of S." [1]

8.4.1 Subcategory

Category (Name, Description, *Parent_Category_Name*)

The 1:N relationship Subcategory is mapped by choosing the participating entity type Category to serve in the role of S, because it is a self-relationship. Moreover, we added the foreign key Parent_Category_Name to connect the entity type to itself.

8.4.2 Contains

Product (SKU, Name, Price, Description, Weight, Brand, Width, Height, Length, *Category_name*)

The 1:N relationship Contains is mapped by choosing the participating entity type Product to serve in the role of S, because its participation in the Contains relationship is from the N-side. Moreover, we added the foreign key Category_name to connect the two participating entities.

8.4.3 Supply

Product (SKU, Name, Price, Description, Weight, Brand, Width, Height, Length, *Category_name*, *Supplier_website*)

The 1:N relationship Supply is mapped by choosing the participating entity type Product to serve in the role of S, because its participation in the Supply relationship is from the N-side. Moreover, we added the foreign key Supplier_website to connect the two participating entities.

8.4.4 Works In

Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email, First_name, Last_name, Phone_number, Country, State, City, Street, Building, Apartment, *Branch_phone_number*)

The 1:N relationship Works_in is mapped by choosing the participating entity type Employee to serve in the role of S, because its participation in the Works_in relationship is from the N-side. Moreover, we added the foreign key Branch_phone_number to connect the two participating entities.

8.4.5 Supervision

Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email, First_name, Last_name, Phone_number, Country, State, City, Street, Building, Apartment, *Branch_phone_number*, *Supervisor_SSN*)

The 1:N relationship Supervision is mapped by choosing the participating entity type Employee to serve in the role of S, because it is a self-relationship. Moreover, we added the foreign key Supervisor_SSN to connect the entity type to itself.

8.4.6 Physical Checkout

Order (Order_id, Notes, Payment_method, Total_amount, Is_online, *Employee_SSN*)

The 1:N relationship Physical_checkout is mapped by choosing the participating entity type Order to serve in the role of S, because its participation in the Physical_checkout relationship is from the N-side. Moreover, we added the foreign key Employee_SSN to connect the two participating entities.

8.4.7 Made By

Order (Order_id, Notes, Payment_method, Total_amount, Is_online, *Employee_SSN*, *Customer_phone_number*, Date)

The 1:N relationship Made_by is mapped by choosing the participating entity type Order to serve in the role of S, because its participation in the Made_by relationship is from the N-side. Moreover, we added the foreign key Customer_phone_number to connect the two participating entities. Finally, we added the atomic attribute of the relationship Made_by to the relation Order.

8.4.8 Works For

Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email, First_name, Last_name, Phone_number, Country, State, City, Street, Building, Apartment, *Branch_phone_number*, *Supervisor_SSN*, *Department_name*)

The 1:N relationship Works_for is mapped by choosing the participating entity type Employee to serve in the role of S, because its participation in the Works_for relationship is from the N-side. Moreover, we added the foreign key Department_name to connect the two participating entities. Finally, we added all the atomic attributes of the relationship Works_for to the relation Employee.

8.4.9 Dependents Of

Dependent (Owner_SSN, Name, Gender, Date_of_birth, Relationship, *Employee_SSN*)

The 1:N relationship Dependents_of is mapped by choosing the participating entity type Dependent to serve in the role of S, because its participation in the Dependents_of relationship is from the N-side. Moreover, we added the foreign key Employee_SSN to connect the two participating entities.

8.4.10 Assigned To

Support_ticket (Ticket_id, Description, Subject, Status, Priority, *Employee_SSN*)

The 1:N relationship Assigned_to is mapped by choosing the participating entity type Support_ticket to serve in the role of S, because its participation in the Assigned_to relationship is from the N-side. Moreover, we added the foreign key Employee_SSN to connect the two participating entities.

8.4.11 Delivers

Order (Order_id, Notes, Payment_method, Total_amount, Is_online, *Employee_SSN*, *Customer_phone_number*, Date, *Driver_license_number*)

The 1:N relationship Delivers is mapped by choosing the participating entity type Order to serve in the role of S, because its participation in the Delivers relationship is from the N-side. Moreover, we added the foreign key Driver_license_number to connect the two participating entities.

8.4.12 Requests

Support_ticket (Ticket_id, Description, Subject, Status, Priority, *Employee_SSN*, *Customer_phone_number*)

The 1:N relationship Requests is mapped by choosing the participating entity type Support_ticket to serve in the role of S, because its participation in the Requests relationship is from the N-side. Moreover, we added the foreign key Custom_phone_number to connect the two participating entities.

8.5 Mapping of Multivalued Attributes

- According to Elmasri and Navathe, in Fundamentals of Database Systems (2015), "For each regular binary M:N relationship type R, create a new relation S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; combination will form the primary key of S.
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S." [1]

8.5.1 Wishlist

Wishlist (SKU, *Customer_phone_number*, Total_amount)

The M:N relationship Wishlist from the ER diagram is mapped by creating a relation Wishlist in the relational database schema. The primary keys of the Product and Customer relations are included as foreign keys in Wishlist and renamed Product_SKU and Customer_phone_number, respectively. The attribute Total_amount represents the total amount of the product in the wish list. The primary key of Wishlist is the tuple of foreign keys {Product_SKU, Customer_phone_number}.

8.5.2 Located In

Located_in (*Product_SKU*, *Branch_phone_number*, Quantity, Shelf_location)

The M:N relationship type Located_in from the ER diagram is mapped by creating a relation Located_in in the relational database schema. The primary keys of the Product and Branch relations are included as foreign keys in Located_in and renamed Product_SKU and Branch_phone_number, respectively. Attributes Quantity and Shelf_location in Located_in represent the Quantity and Shelf_location attributes of the relation type. The primary key of the Located_in relation is the combination of the foreign key attributes {Product_SKU, Branch_phone_number}.

8.5.3 Reviews

Reviews (*Product_SKU*, *Customer_phone_number*, Review_date, Rating, Comment, Description)

The M:N relationship type Reviews from the ER diagram is mapped by creating a relation Reviews in the relational database schema. The primary keys of the Product and Customer relations are included as foreign keys in Reviews and renamed Product_SKU and Customer_phone_number, respectively. Attributes Review_date, Rating, Comment, and Description in Reviews represent the corresponding attributes of the relation type. The primary key of the Reviews relation is the combination of the foreign key attributes {Product_SKU, Customer_phone_number}. Furthermore, we didn't include the multi-valued attribute Image_URLs as we'll create its own relation later.

8.5.4 Purchased

Purchased (*Product_SKU*, *Order_id*, Quantity, Amount)

The M:N relationship type Purchased from the ER diagram is mapped by creating a relation Purchased in the relational database schema. The primary keys of the Product and Order relations are included as foreign keys in Purchased and renamed Product_SKU and Order_id, respectively. Attributes Quantity and Amount in Purchased represent the corresponding attributes of the relation type. The primary key of the Purchased relation is the combination of the foreign key attributes {Product_SKU, Order_id}.

8.6 Mapping of Binary M:N Relationship Types

- According to Elmasri and Navathe, in Fundamentals of Database Systems (2015), "For each multivalued attribute A, create a new relation R.

- This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components." [1]

8.6.1 Colors

Colors (*Product_SKU*, Product_color)

The relation Colors is created. The attribute Product_color represents the multivalued attribute Colors of Product, while Product_SKU—as foreign key—represents the primary key of the Product relation. The primary key of Color_s is the combination of {Product_SKU, Product_color}.

8.6.2 Image URLs

Image_URLs (*Product_SKU*, *Customer_phone_number*, Product_Image_URL)

The relation Image_URLs is created. The attribute Product_Image_URL represents the multivalued attribute Image_URL of Product, while Product_SKU—as foreign key—represents the primary key of the Product relation. The primary key of Image_URLs is the combination of {Product_SKU, Product_Image_URL}.

8.6.3 Working Hours

Working_hours (*Branch_phone_number*, Day, Opening_hour, Closing_hour)

The relation Working_hours is created. The attributes Day, Opening_hour, and Closing_hour represent the composite-multivalued attribute Work_hours of Branch, while Branch_phone_number—as foreign key—represents the primary key of the Branch relation. The primary key of Working_hours is the combination of {Branch_phone_number, Day, Opening_hour, Closing_hour}.

8.6.4 Department Location

Department_location (Department_name, Location)

The relation Department_location is created. The attribute Location represents the multivalued attribute Locations of Department, while Department_name—as foreign key—represents the primary key of the Department relation. The primary key of Department_location is the combination of {Department_name, Location}.

9 Final Display – All Tables

Table 9.1: *Branch Table*

Phone_number	Name	Country	State	City	Street	Building	Apartment	*Employee_SSN*

Table 9.2: *Category Table*

Name	Description	*Parent_Category_Name*

Table 9.3: *Colors Table*

Product_SKU	Product_color

Table 9.4: *Coupon Table*

Code	Description	Discount_percent	Times_used	Minimum_order_amount	Maximum_order_amount	Usage_limit	Valid_from	Valid_to	*Order_ID*	Discount_amount	Redeem_date

Table 9.5: *Customer Table*

Phone_number	Email	First_name	Last_name	Gender	Registration_date	Password_hashed	Date_of_birth	Country	State	City	Street	Building	Apartment

Table 9.6: *Department Table*

Name	Number_of_employees	*Employee_SSN*	Manager_start_date

Table 9.7: *Department Location Table*

Department_name	Location

Table 9.8: *Dependent Table*

Employee_SSN	Name	Gender	Date_of_birth	Relationship

Table 9.9: *Driver Table*

License_number	Driving_experience_years	License_expiry_date	*Employee_SSN*

Table 9.10: *Employee Table*

SSN	Position	Salary	Hire_date	Gender	Date_of_birth	Email	First_name	Last_name	Phone_number	Country	State	City	Street	Building	Apartment	*Branch_phone_number*	*Supervisor_SSN*	*Department_name*

Table 9.11: *Image URLs Table*

Product_SKU	*Customer_phone_number*	Product_Image_URL

Table 9.12: *Located In Table*

Product_SKU	*Branch_phone_number*	Quantity	Shelf_location

Table 9.13: *Order Table*

Order_id	Notes	Payment_method	Total_amount	Is_online	*Employee_SSN*	*Customer_phone_number*	Date	*Driver_license_number*

Table 9.14: *Product Table*

SKU	Name	Price	Description	Weight	Brand	Width	Height	Length	*Category_name*	*Supplier_website*

Table 9.15: *Purchased Table*

Product_SKU	*Order_id*	Quantity	Amount

Table 9.16: *Reviews Table*

Product_SKU	*Customer_phone_number*	Review_date	Rating	Comment	Description

Table 9.17: *Support Ticket Table*

Ticket_id	Description	Subject	Status	Priority	*Employee_SSN*	*Customer_phone_number*

Table 9.18: *Supplier Table*

Website	Supplier_name	Contact_person_email	Contact_person_first_name	Contact_person_last_name	Contact_person_phone_number

Table 9.19: *Wishlist Table*

SKU	*Customer_phone_number*	Total_amount

Table 9.20: *Working Hours Table*

Branch_phone_number	Day	Opening_hour	Closing_hour

10 Tables' States

Table 10.1: *Branch*

Phone_number	Name	Country	State	City	Street	Building	Apartment	*Employee_SSN*
+96111111111	Beirut Main	Lebanon	Beirut	Beirut	Hamra	10	1	123456789
+96122222222	Tripoli Branch	Lebanon	North	Tripoli	Mina Street	5	12	987654321
+96133333333	Sidon Hub	Lebanon	South	Sidon	Corniche	3	6	543216789
+96144444444	Zahle Branch	Lebanon	Beqaa	Zahle	Bekaa St	8	4	123459876
+96155555555	Jounieh Store	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	7	3	567894321
+96166666666	Byblos Outlet	Lebanon	Mount Lebanon	Byblos	Roman Street	6	5	678912345
+96177777777	Tyre Shop	Lebanon	South	Tyre	Port Road	2	1	876543219
+96188888888	Baalbek Point	Lebanon	Beqaa	Baalbek	Temple Rd	4	2	345678912
+96199999999	Batroun Corner	Lebanon	North	Batroun	Old City	1	1	456789123
+96112345678	Downtown Center	Lebanon	Beirut	Beirut	Downtown	5	11	234567891
+96198765432	Achrafieh Depot	Lebanon	Beirut	Achrafieh	Armenia St	9	8	234567891
+96124681357	Dora Warehouse	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	2	789123456
+96165432198	Aley Branch	Lebanon	Mount Lebanon	Aley	Souk Street	4	10	654321987
+96111223344	Choueifat Station	Lebanon	Mount Lebanon	Choueifat	Railway Rd	2	7	321987654
+96144332211	Antelias Spot	Lebanon	Mount Lebanon	Antelias	Highway Rd	8	5	987123456

Table 10.2: *Category*

Name	Description	*Parent_Category_Name*
Electronics	Devices and gadgets	NULL
Clothing	Apparel and fashion items	NULL
Furniture	Home and office furniture	NULL
Books	Printed and digital books	Stationery
Groceries	Food and daily supplies	Health
Sports	Sporting equipment and apparel	Clothing
Beauty	Cosmetics and skincare products	Health
Toys	Toys for kids and adults	Sports
Automotive	Car parts and accessories	Electronics
Jewelry	Watches, rings, and necklaces	Fashion
Health	Medical supplies and equipment	Beauty
Stationery	Office and school supplies	Furniture
Pets	Pet food and accessories	Groceries
Music	Instruments and music equipment	Art
Art	Art supplies and crafts	Stationery

Table 10.3: *Colors*

Product_SKU	Product_color
1001	Red
1002	Blue
1003	Green
1004	Black
1005	White
1006	Yellow
1007	Pink
1008	Purple
1009	Orange
1010	Brown
1011	Gray
1012	Gold
1013	Silver
1014	Beige
1015	Navy

Table 10.4: *Coupon*

Code	Description	Discount_percent	Times_used	Minimum_order_amount	Maximum_order_amount	Usage_limit	Valid_from	Valid_to	*Order_ID*	Discount_amount	Redeem_date
DIS10	10% off	10	25	50	500	100	2024-01-01	2024-12-31	O101	5	2024-10-01
DIS20	20% off	20	40	100	1000	50	2024-02-01	2024-11-30	O102	10	2024-09-15
FREESHIP	Free shipping	0	100	0	200	200	2024-05-01	2024-10-31	O103	0	2024-08-12
SAVE15	15% off	15	60	75	750	75	2024-03-01	2024-09-30	O104	12	2024-07-07
BOGO	Buy 1 Get 1	50	20	100	1000	25	2024-01-01	2024-08-31	O105	50	2024-06-06
HOLIDAY50	50% off	50	10	200	2000	10	2024-12-01	2024-12-31	O106	100	2024-12-05
FLASH5	5% off	5	150	25	250	500	2024-07-01	2024-08-01	O107	2	2024-07-12
SUMMER30	30% off	30	80	150	1500	30	2024-06-01	2024-09-01	O108	45	2024-07-20
BLACKFRIDAY	40% off	40	90	300	3000	100	2024-11-29	2024-11-29	O109	120	2024-11-29
NEWYEAR25	25% off	25	70	200	2000	75	2024-12-31	2025-01-01	O110	50	2024-12-31
WELCOME	10% for new users	10	200	50	500	300	2024-01-01	2024-12-31	O111	10	2024-01-10
VIP20	20% VIP discount	20	30	150	1500	50	2024-01-01	2024-12-31	O112	30	2024-02-14
BIRTHDAY	25% off on birthday	25	10	100	1000	20	2024-01-01	2024-12-31	O113	25	2024-03-03
CLEARANCE	Up to 70% off	70	5	500	5000	10	2024-11-01	2024-11-15	O114	350	2024-11-05
LOYALTY	15% off for loyal customers	15	50	100	1000	60	2024-01-01	2024-12-31	O115	15	2024-04-22

Table 10.5: *Customer*

Phone_number	Email	First_name	Last_name	Gender	Registration_date	Password_hashed	Date_of_birth	Country	State	City	Street	Building	Apartment
+96134567890	john.doe@gmail.com	John	Doe	Male	2023-02-14	********	1990-03-05	Lebanon	Beirut	Beirut	Hamra	12	2
+96198765432	jane.smith@yahoo.com	Jane	Smith	Female	2022-10-22	********	1985-06-15	Lebanon	North	Tripoli	Mina Street	4	5
+96145678901	alice.brown@outlook.com	Alice	Brown	Female	2024-03-01	********	1992-01-20	Lebanon	South	Sidon	Corniche	7	3
+96167890123	bob.jones@hotmail.com	Bob	Jones	Male	2021-07-18	********	1988-11-02	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	8	6
+96178901234	charlie.evans@gmail.com	Charlie	Evans	Male	2020-05-30	********	1995-09-09	Lebanon	Beirut	Achrafieh	Armenia St	9	7
+96189012345	diana.lee@aol.com	Diana	Lee	Female	2023-01-15	********	1989-04-25	Lebanon	Beqaa	Zahle	Bekaa St	2	1
+96190123456	frank.wilson@protonmail.com	Frank	Wilson	Male	2019-11-20	********	1975-12-12	Lebanon	North	Batroun	Old City	1	1
+96101234567	emma.white@gmail.com	Emma	White	Female	2024-06-01	********	1993-08-17	Lebanon	Mount Lebanon	Byblos	Roman Street	6	3
+96123456789	george.king@live.com	George	King	Male	2022-09-05	********	1980-02-22	Lebanon	South	Tyre	Port Road	5	10
+96156789012	hannah.scott@gmail.com	Hannah	Scott	Female	2021-12-25	********	2000-07-30	Lebanon	Mount Lebanon	Antelias	Highway Rd	8	9
+96167890124	jack.miller@yahoo.com	Jack	Miller	Male	2023-08-10	********	1987-05-15	Lebanon	Beqaa	Baalbek	Temple Rd	4	2
+96178901235	isabella.taylor@outlook.com	Isabella	Taylor	Female	2024-02-02	********	1991-03-18	Lebanon	Beirut	Downtown	Downtown	3	11
+96189012346	kevin.martin@hotmail.com	Kevin	Martin	Male	2022-07-19	********	1984-09-25	Lebanon	Mount Lebanon	Aley	Souk Street	4	10
+96190123457	laura.harris@icloud.com	Laura	Harris	Female	2021-04-22	********	1996-11-05	Lebanon	Beirut	Hamra	Hamra St	2	8
+96101234568	mike.anderson@protonmail.com	Mike	Anderson	Male	2023-05-03	********	1998-01-11	Lebanon	Mount Lebanon	Choueifat	Railway Rd	7	3

Table 10.6: *Department*

Name	Number_of_employees	*Employee_SSN*	Manager_start_date
Sales	30	123456789	2022-03-01
Marketing	25	987654321	2021-06-15
HR	15	123459876	2023-01-10
Finance	20	567894321	2019-10-05
Operations	35	543216789	2020-08-25
IT	10	678912345	2024-04-18
Customer Support	12	876543219	2022-09-12
Logistics	18	345678912	2020-11-20
Legal	8	456789123	2021-02-14
R&D	14	234567891	2018-12-22
Training	9	789123456	2023-03-27
Procurement	11	654321987	2021-05-30
Admin	7	321987654	2022-07-04
Facilities	6	987123456	2019-06-01
Compliance	5	654789321	2020-01-17

Table 10.7: *Department location*

Department_name	Location
Sales	Beirut
Marketing	Tripoli
HR	Zahle
Finance	Jounieh
Operations	Sidon
IT	Byblos
Customer Support	Tyre
Logistics	Baalbek
Legal	Batroun
R&D	Achrafieh
Training	Dora
Procurement	Aley
Admin	Choueifat
Facilities	Antelias
Compliance	Downtown Beirut

Table 10.8: *Dependent*

Employee_SSN	Name	Gender	Date_of_birth	Relationship
123456789	Sarah Doe	Female	2015-04-15	Daughter
987654321	James Smith	Male	2013-07-20	Son
123459876	Emily Brown	Female	2017-02-28	Daughter
567894321	Lucas Jones	Male	2018-11-05	Son
543216789	Olivia Evans	Female	2016-09-12	Daughter
678912345	Ethan White	Male	2020-03-22	Son
876543219	Chloe Lee	Female	2014-08-01	Daughter
345678912	Liam Harris	Male	2015-12-15	Son
456789123	Mia Taylor	Female	2018-10-03	Daughter
234567891	Noah Wilson	Male	2021-06-08	Son
789123456	Sophia Martin	Female	2019-05-25	Daughter
654321987	Benjamin Scott	Male	2012-01-19	Son
321987654	Emma Anderson	Female	2015-07-13	Daughter
987123456	Mason Miller	Male	2018-03-09	Son
654789321	Ava Thomas	Female	2016-02-04	Daughter

Table 10.9: *Driver*

License_number	Driving_experience_years	License_expiry_date	*Employee_SSN*
DL1001	5	2025-12-31	123456789
DL1002	3	2026-06-30	987654321
DL1003	10	2027-04-15	567894321
DL1004	7	2028-01-10	543216789
DL1005	2	2026-08-20	678912345
DL1006	6	2024-11-25	876543219
DL1007	8	2029-09-09	345678912
DL1008	12	2025-05-05	456789123
DL1009	4	2024-07-18	234567891
DL1010	15	2030-03-30	789123456
DL1011	1	2024-12-15	654321987
DL1012	9	2027-02-14	321987654
DL1013	14	2031-06-11	987123456
DL1014	11	2029-12-21	654789321
DL1015	13	2026-10-07	123459876

Table 10.10: *Employee*

SSN	Position	Salary	Hire_date	Gender	Date_of_birth	Email	First_name	Last_name	Phone_number	Country	State	City	Street	Building	Apartment	*Branch_phone_number*	*Supervisor_SSN*	*Department_name*
123456789	Manager	3000	2022-03-01	Male	1985-05-12	john.doe@example.com	John	Doe	+96134567890	Lebanon	Beirut	Beirut	Hamra	12	2	+96111111111	NULL	Sales
987654321	Marketing Head	2800	2021-06-15	Female	1988-08-23	jane.smith@example.com	Jane	Smith	+96198765432	Lebanon	North	Tripoli	Mina St	5	3	+96122222222	123456789	Marketing
123459876	HR Manager	2700	2023-01-10	Male	1990-02-28	bob.jones@example.com	Bob	Jones	+96167890123	Lebanon	Beqaa	Zahle	Bekaa St	2	1	+96144444444	123456789	HR
567894321	Finance Manager	3200	2019-10-05	Female	1986-11-09	alice.brown@example.com	Alice	Brown	+96145678901	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	7	2	+96155555555	123456789	Finance
543216789	Operations Head	3500	2020-08-25	Male	1978-07-05	david.evans@example.com	David	Evans	+96178901234	Lebanon	South	Sidon	Corniche	6	3	+96133333333	567894321	Operations
678912345	IT Specialist	2000	2024-04-18	Female	1995-03-18	emma.white@example.com	Emma	White	+96101234567	Lebanon	Mount Lebanon	Byblos	Roman St	5	4	+96166666666	567894321	IT
876543219	Support Manager	2500	2022-09-12	Male	1983-09-27	frank.wilson@example.com	Frank	Wilson	+96190123456	Lebanon	South	Tyre	Port Rd	3	2	+96177777777	543216789	Customer Support
345678912	Logistics Head	3000	2020-11-20	Female	1989-04-15	olivia.harris@example.com	Olivia	Harris	+96156789012	Lebanon	Beqaa	Baalbek	Temple Rd	4	3	+96188888888	543216789	Logistics
456789123	Legal Advisor	2900	2021-02-14	Male	1982-12-12	george.king@example.com	George	King	+96123456789	Lebanon	North	Batroun	Old City	2	1	+96199999999	567894321	Legal
234567891	R&D Manager	3100	2018-12-22	Female	1987-06-06	sophia.martin@example.com	Sophia	Martin	+96178901235	Lebanon	Beirut	Achrafieh	Armenia St	8	2	+96112345678	543216789	R&D
789123456	Trainer	2200	2023-03-27	Male	1991-09-30	kevin.martin@example.com	Kevin	Martin	+96189012346	Lebanon	Mount Lebanon	Dora	Industrial Zone	5	2	+96124681357	234567891	Training
654321987	Procurement Officer	2400	2021-05-30	Female	1993-01-14	laura.harris@example.com	Laura	Harris	+96167890124	Lebanon	Mount Lebanon	Aley	Souk St	4	10	+96165432198	234567891	Procurement
321987654	Admin Assistant	1800	2022-07-04	Male	1996-07-19	jack.miller@example.com	Jack	Miller	+96101234568	Lebanon	Mount Lebanon	Choueifat	Railway Rd	2	8	+96111223344	789123456	Admin
987123456	Facilities Manager	2700	2019-06-01	Female	1979-03-02	diana.lee@example.com	Diana	Lee	+96189012345	Lebanon	Mount Lebanon	Antelias	Highway Rd	9	5	+96144332211	789123456	Facilities
654789321	Compliance Officer	2600	2020-01-17	Male	1984-11-04	mike.anderson@example.com	Mike	Anderson	+96190123457	Lebanon	Beirut	Downtown	Downtown	6	3	+96111223344	789123456	Compliance

Table 10.11: *Image URLs*

Product_SKU	*Customer_phone_number*	Product_Image_URL
1001	+96134567890	https://example.com/image1.jpg
1002	+96198765432	https://example.com/image2.jpg
1003	+96145678901	https://example.com/image3.jpg
1004	+96167890123	https://example.com/image4.jpg
1005	+96178901234	https://example.com/image5.jpg
1006	+96189012345	https://example.com/image6.jpg
1007	+96190123456	https://example.com/image7.jpg
1008	+96101234567	https://example.com/image8.jpg
1009	+96123456789	https://example.com/image9.jpg
1010	+96156789012	https://example.com/image10.jpg
1011	+96167890124	https://example.com/image11.jpg
1012	+96178901235	https://example.com/image12.jpg
1013	+96189012346	https://example.com/image13.jpg
1014	+96190123457	https://example.com/image14.jpg
1015	+96101234568	https://example.com/image15.jpg

Table 10.12: *Located in*

Product_SKU	*Branch_phone_number*	Quantity	Shelf_location
1001	+96111111111	50	A1
1002	+96122222222	30	B2
1003	+96133333333	20	C3
1004	+96144444444	15	D4
1005	+96155555555	60	E5
1006	+96166666666	25	F6
1007	+96177777777	10	G7
1008	+96188888888	35	H8
1009	+96199999999	40	I9
1010	+96112345678	50	J10
1011	+96198765432	70	K11
1012	+96124681357	20	L12
1013	+96165432198	15	M13
1014	+96111223344	5	N14
1015	+96144332211	55	O15

Table 10.13: *Order*

Order_id	Notes	Payment_method	Total_amount	Is_online	*Employee_SSN*	*Customer_phone_number*	Date	*Driver_license_number*
O101	Expedited delivery	Credit Card	150	Yes	123456789	+96134567890	DL1001	2024-10-01
O102	Gift wrap included	Cash on Delivery	200	No	987654321	+96198765432	DL1002	2024-09-15
O103	Deliver before 5 PM	PayPal	75	Yes	567894321	+96145678901	DL1003	2024-08-12
O104	Call on arrival	Credit Card	300	No	543216789	+96167890123	DL1004	2024-07-07
O105	Special instructions	Apple Pay	500	Yes	678912345	+96178901234	DL1005	2024-06-06
O106	Holiday gift	Credit Card	1000	No	876543219	+96189012345	DL1006	2024-12-05
O107	Contactless delivery	PayPal	250	Yes	345678912	+96190123456	DL1007	2024-07-12
O108	Scheduled for 3 PM	Credit Card	150	Yes	456789123	+96101234567	DL1008	2024-07-20
O109	Express delivery	Cash on Delivery	500	No	234567891	+96123456789	DL1009	2024-11-29
O110	New Year's package	Apple Pay	750	Yes	789123456	+96156789012	DL1010	2024-12-31
O111	First-time discount	PayPal	120	Yes	654321987	+96167890124	DL1011	2024-01-10
O112	VIP priority	Credit Card	800	No	321987654	+96178901235	DL1012	2024-02-14
O113	Happy Birthday!	Apple Pay	180	Yes	987123456	+96189012346	DL1013	2024-03-03
O114	Clearance sale	PayPal	450	No	654789321	+96190123457	DL1014	2024-11-05
O115	Loyalty customer	Cash on Delivery	300	No	123459876	+96101234568	DL1015	2024-04-22

Table 10.14: *Product*

SKU	Name	Price	Description	Weight	Brand	Width	Height	Length	*Category_name*	*Supplier_website*
1001	Smartphone	600	5G-enabled phone	200g	TechBrand	7cm	15cm	0.8cm	Electronics	www.techbrand.com
1002	Laptop	1200	Ultrabook with 16GB RAM	1.5kg	ComputeX	32cm	22cm	1.5cm	Electronics	www.computex.com
1003	Office Chair	150	Ergonomic chair	12kg	ComfortCo	60cm	120cm	60cm	Furniture	www.comfortco.com
1004	Running Shoes	100	Lightweight shoes	500g	SportWear	12cm	35cm	10cm	Sports	www.sportwear.com
1005	Acoustic Guitar	300	6-string guitar	3kg	MusicPro	38cm	100cm	12cm	Music	www.musicpro.com
1006	Refrigerator	800	Double door fridge	65kg	HomeAppl	90cm	180cm	75cm	Electronics	www.homeappl.com
1007	LED TV	400	50-inch 4K UHD	8kg	VisionCo	112cm	65cm	5cm	Electronics	www.visionco.com
1008	Blender	70	High-speed blender	2kg	KitchenX	20cm	40cm	15cm	Electronics	www.kitchenx.com
1009	T-Shirt	25	Cotton T-shirt	250g	FashionHub	30cm	80cm	1cm	Clothing	www.fashionhub.com
1010	Watch	500	Smartwatch with GPS	200g	TimeKeep	5cm	5cm	1cm	Jewelry	www.timekeep.com
1011	Textbook	50	Advanced mathematics book	1kg	EduBooks	21cm	28cm	3cm	Books	www.edubooks.com
1012	Desk Lamp	30	LED desk lamp	1.5kg	LightPro	15cm	40cm	15cm	Furniture	www.lightpro.com
1013	Wireless Headphones	150	Noise-canceling headphones	300g	AudioMax	18cm	20cm	5cm	Electronics	www.audiomax.com
1014	Soccer Ball	40	FIFA approved	450g	SportWear	22cm	22cm	22cm	Sports	www.sportwear.com
1015	Gaming Console	500	Next-gen console	3kg	GameZone	30cm	10cm	25cm	Electronics	www.gamezone.com

Table 10.15: *Purchased*

Product_SKU	*Order_id*	Quantity	Amount
1001	O101	2	1200
1002	O102	1	1200
1003	O103	1	150
1004	O104	2	200
1005	O105	1	300
1006	O106	1	800
1007	O107	1	400
1008	O108	3	210
1009	O109	5	125
1010	O110	2	1000
1011	O111	1	50
1012	O112	2	60
1013	O113	1	150
1014	O114	3	120
1015	O115	1	500

Table 10.16: *Reviews*

Product_SKU	*Customer_phone_number*	Review_date	Rating	Comment	Description
1001	+96134567890	2024-09-01	5	Great phone!	Excellent performance
1002	+96198765432	2024-08-20	4	Good value	Worth the price
1003	+96145678901	2024-07-10	3	Comfortable chair	Could be sturdier
1004	+96167890123	2024-06-05	5	Love these shoes	Perfect fit
1005	+96178901234	2024-05-15	4	Nice sound	Great for beginners
1006	+96189012345	2024-04-22	5	Amazing fridge	Lots of space
1007	+96190123456	2024-03-30	4	Clear picture	Excellent for gaming
1008	+96101234567	2024-02-18	3	Good blender	Noisy motor
1009	+96123456789	2024-01-11	5	Comfortable T-shirt	Soft fabric
1010	+96156789012	2024-09-25	4	Nice smartwatch	Battery life could be better
1011	+96167890124	2024-08-05	5	Informative book	Highly recommended
1012	+96178901235	2024-07-22	4	Useful lamp	Bright and adjustable
1013	+96189012346	2024-06-12	5	Excellent headphones	Superb sound quality
1014	+96190123457	2024-05-02	4	Great soccer ball	Durable material
1015	+96101234568	2024-03-28	5	Fantastic console	Best for gaming enthusiasts

Table 10.17: *Support Ticket*

Ticket_id	Description	Subject	Status	Priority	*Employee_SSN*	*Customer_phone_number*
T001	Issue with product delivery	Delivery Issue	Open	High	123456789	+96134567890
T002	Request for refund	Refund Request	Closed	Medium	987654321	+96198765432
T003	Product not working	Defective Product	Open	High	567894321	+96145678901
T004	Inquiry about order status	Order Inquiry	Resolved	Low	543216789	+96167890123
T005	Delayed shipment	Shipment Delay	Open	Medium	678912345	+96178901234
T006	Cancel order request	Order Cancellation	Closed	Medium	876543219	+96189012345
T007	Issue with payment	Payment Issue	Resolved	High	345678912	+96190123456
T008	Exchange request	Product Exchange	Open	Medium	456789123	+96101234567
T009	Warranty inquiry	Warranty Inquiry	Resolved	Low	234567891	+96123456789
T010	Complaint about service	Service Complaint	Open	High	789123456	+96156789012
T011	Missing items in order	Missing Items	Open	Medium	654321987	+96167890124
T012	Subscription issue	Subscription Problem	Resolved	Low	321987654	+96178901235
T013	Wrong product delivered	Wrong Product	Open	High	987123456	+96189012346
T014	Feedback submission	Customer Feedback	Closed	Low	654789321	+96190123457
T015	Request for discount	Discount Request	Open	Medium	123459876	+96101234568

Table 10.18: *Supplier*

Website	Supplier_name	Contact_person_email	Contact_person_first_name	Contact_person_last_name	Contact_person_phone_number
www.techbrand.com	TechBrand	contact@techbrand.com	Alice	Doe	+96134567890
www.computex.com	ComputeX	support@computex.com	John	Smith	+96198765432
www.comfortco.com	ComfortCo	info@comfortco.com	Emma	Johnson	+96145678901
www.sportwear.com	SportWear	sales@sportwear.com	Michael	Brown	+96167890123
www.musicpro.com	MusicPro	music@musicpro.com	Sarah	Lee	+96178901234
www.homeappl.com	HomeAppl	appliances@homeappl.com	David	Evans	+96189012345
www.visionco.com	VisionCo	vision@visionco.com	Jessica	Taylor	+96190123456
www.kitchenx.com	KitchenX	service@kitchenx.com	Kevin	Wilson	+96101234567
www.fashionhub.com	FashionHub	hello@fashionhub.com	Emily	Harris	+96123456789
www.timekeep.com	TimeKeep	contact@timekeep.com	Daniel	Martin	+96156789012
www.edubooks.com	EduBooks	edu@edubooks.com	Olivia	White	+96167890124
www.lightpro.com	LightPro	light@lightpro.com	Robert	Scott	+96178901235
www.audiomax.com	AudioMax	audio@audiomax.com	Sophia	Anderson	+96189012346
www.gamezone.com	GameZone	games@gamezone.com	Liam	Thomas	+96190123457
www.sportwear.com	SportWear	store@sportwear.com	Noah	Miller	+96101234568

Table 10.19: *Wishlist*

SKU	*Customer_phone_number*	Total_amount
1001	+96134567890	600
1002	+96198765432	1200
1003	+96145678901	150
1004	+96167890123	100
1005	+96178901234	300
1006	+96189012345	800
1007	+96190123456	400
1008	+96101234567	210
1009	+96123456789	125
1010	+96156789012	1000
1011	+96167890124	50
1012	+96178901235	60
1013	+96189012346	150
1014	+96190123457	120
1015	+96101234568	500

Table 10.20: *Working hours*

Branch_phone_number	Day	Opening_hour	Closing_hour
+96111111111	Monday	09:00	18:00
+96122222222	Tuesday	09:00	18:00
+96133333333	Wednesday	09:00	18:00
+96144444444	Thursday	09:00	18:00
+96155555555	Friday	09:00	18:00
+96166666666	Saturday	09:00	14:00
+96177777777	Sunday	Closed	Closed
+96188888888	Monday	09:00	18:00
+96199999999	Tuesday	09:00	18:00
+96112345678	Wednesday	09:00	18:00
+96198765432	Thursday	09:00	18:00
+96124681357	Friday	09:00	18:00
+96165432198	Saturday	09:00	14:00
+96111223344	Sunday	Closed	Closed
+96144332211	Monday	09:00	18:00

11 SQL DDL

11.1 Queries

11.1.1 Create Queries

First of all, let us define the domains.

```
1 CREATE DOMAIN VC AS VARCHAR(50);
2
3 CREATE DOMAIN PHONE_NUMBER AS VARCHAR(25);
4
5 CREATE DOMAIN ID AS CHAR(10);
6
7 CREATE DOMAIN TEXT AS VARCHAR(255);
```

Code Snippet 11.1 : *Create Domains*

11.1.1.1 Supplier

The following script creates the Supplier table.

```
CREATE TABLE
  IF NOT EXISTS Supplier (
    Website VC NOT NULL,
    Supplier_name VC NOT NULL,
    Contact_person_email VC NOT NULL,
    Contact_person_first_name VC NOT NULL,
    Contact_person_last_name VC NOT NULL,
    Contact_person_phone_number PHONE_NUMBER NOT NULL,
    CONSTRAINT PK_Supplier PRIMARY KEY (Website),
    CONSTRAINT Supplier_CK_Contact_person_email CHECK (Contact_person_email
        LIKE '%@%.%'),
    CONSTRAINT Supplier_UK_Supplier_name UNIQUE (Supplier_name),
    CONSTRAINT Supplier_UK_Contact_person_phone_number UNIQUE (
       Contact_person_phone_number),
    CONSTRAINT Supplier_UK_Contact_person_email UNIQUE (
       Contact_person_email)
  );
```

Code Snippet 11.2 : *Create Supplier Table*

11.1.1.2 Category

The following script creates the Category table.

```
CREATE TABLE
  IF NOT EXISTS Category (
    Name VC NOT NULL,
    Description TEXT NOT NULL,
    Parent_Category_Name VC,
    CONSTRAINT PK_Category PRIMARY KEY (Name),
    CONSTRAINT Category_FK_Category FOREIGN KEY (Parent_Category_Name)
       REFERENCES Category (Name) ON UPDATE CASCADE ON DELETE SET NULL
  );
```

Code Snippet 11.3 : *Create Category Table*

11.1.1.3 Product

The following script creates the Product table.

```
CREATE TABLE
  IF NOT EXISTS Product (
    SKU VC NOT NULL,
    Name VC NOT NULL,
    Price INT NOT NULL,
    Description TEXT,
```

```
    Weight INT,
    Brand VC,
    Width INT,
    Height INT,
    Length INT,
    Category_name VC,
    Supplier_website TEXT,
    CONSTRAINT PK_PRODUCT PRIMARY KEY (SKU),
    CONSTRAINT Product_FK_Category_name FOREIGN KEY (Category_name)
        REFERENCES Category (Name) ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Product_FK_Supplier_website FOREIGN KEY (Supplier_website)
        REFERENCES Supplier (Website) ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Product_CK_Price CHECK (Price > 0),
    CONSTRAINT Product_CK_Weight CHECK (Weight > 0),
    CONSTRAINT Product_CK_Height CHECK (Height > 0),
    CONSTRAINT Product_CK_Length CHECK (Length > 0),
    CONSTRAINT Product_UK_Product UNIQUE (Name, Category_name),
    CONSTRAINT Product_UK_Supplier UNIQUE (Supplier_website, SKU)
  );
```

Code Snippet 11.4 : *Create Product Table*

11.1.1.4 Branch

The following script creates the Branch table.

```
CREATE TABLE
  IF NOT EXISTS Branch (
    Phone_number PHONE_NUMBER NOT NULL,
    Name VC NOT NULL,
    Country VC NOT NULL,
    State VC NOT NULL,
    City VC NOT NULL,
    Street VC NOT NULL,
    Building INT NOT NULL,
    Apartment INT NOT NULL,
    Employee_SSN VC,
    CONSTRAINT PK_Branch PRIMARY KEY (Phone_number),
    CONSTRAINT Branch_CK_Apartment CHECK (Apartment >= 0),
    CONSTRAINT Branch_CK_Building CHECK (Building >= 0),
    CONSTRAINT Branch_UK_LOCATION UNIQUE (Country, State, City, Street,
        Building, Apartment),
    CONSTRAINT Branch_UK_EMPLOYEE UNIQUE (Employee_SSN)
  );
```

Code Snippet 11.5 : *Create Branch Table*

11.1.1.5 Department

The following script creates the Department table.

```
CREATE TABLE
  IF NOT EXISTS Department (
    Name VC NOT NULL,
    Number_of_employees INT NOT NULL,
    Employee_SSN VC,
    Manager_start_date DATE NOT NULL,
    CONSTRAINT PK_Department PRIMARY KEY (Name),
    CONSTRAINT Department_Department_CK_Number_of_employees CHECK (
        Number_of_employees > 0),
    CONSTRAINT Department_Department_CK_Manager_start_date CHECK (
        Manager_start_date < CURRENT_DATE),
    CONSTRAINT Department_Department_UK_Manager UNIQUE (Employee_SSN),
    CONSTRAINT Department_Department_UK_Department UNIQUE (Name)
  );
```

Code Snippet 11.6 : *Create Department Table*

11.1.1.6 Employee

The following script creates the Employee table.

```
CREATE TABLE
  IF NOT EXISTS Employee (
    SSN VC NOT NULL,
    Position VC NOT NULL,
    Salary INT NOT NULL,
    Hire_date DATE NOT NULL,
    Gender VARCHAR(6) NOT NULL,
    Date_of_birth DATE NOT NULL,
    Email VC NOT NULL,
    First_name VC NOT NULL,
    Last_name VC NOT NULL,
    Phone_number PHONE_NUMBER NOT NULL,
    Country VC NOT NULL,
    State VC NOT NULL,
    City VC NOT NULL,
    Street VC NOT NULL,
    Building INT NOT NULL,
    Apartment INT NOT NULL,
    Branch_phone_number PHONE_NUMBER,
    Supervisor_SSN VC,
    Department_name VC NOT NULL,
    CONSTRAINT PK_EMPLOYEE PRIMARY KEY (SSN),
    CONSTRAINT Employee_FK_Branch_phone_number FOREIGN KEY (
        Branch_phone_number) REFERENCES Branch (Phone_number) ON UPDATE
        CASCADE ON DELETE SET NULL,
```

```
    CONSTRAINT Employee_FK_Supervisor_SSN FOREIGN KEY (Supervisor_SSN)
        REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Employee_FK_Department_name FOREIGN KEY (Department_name)
        REFERENCES Department (Name) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Employee_CK_Gender CHECK (Gender IN ('Male', 'Female')),
    CONSTRAINT Employee_CK_Date_of_birth CHECK (Date_of_birth <
        CURRENT_DATE),
    CONSTRAINT Employee_CK_Salary CHECK (Salary > 0),
    CONSTRAINT Employee_CK_Apartment CHECK (Apartment >= 0),
    CONSTRAINT Employee_CK_Building CHECK (Building >= 0),
    CONSTRAINT Employee_CK_Hire_date CHECK (Hire_date < CURRENT_DATE),
    CONSTRAINT Employee_CK_Email CHECK (Email LIKE '%@%.%'),
    CONSTRAINT Employee_CK_Position CHECK (
      Position IN ('Manager', 'MarketingHead', 'HRManager', 'FinanceManager
          ', 'OperationsHead', 'ITSpecialist', 'SupportManager', '
          LogisticsHead', 'LegalAdvisor', 'R&DManager', 'Trainer', '
          ProcurementOfficer', 'AdminAssistant', 'FacilitiesManager', '
          ComplianceOfficer', 'Driver')
    ),
    CONSTRAINT Employee_UK_Phone_number UNIQUE (Phone_number),
    CONSTRAINT Employee_UK_Email UNIQUE (Email),
    CONSTRAINT Employee_UK_SSN UNIQUE (SSN),
    CONSTRAINT Employee_UK_LOCATION UNIQUE (Country, State, City, Street,
        Building, Apartment)
  );
```

Code Snippet 11.7 : *Create Employee Table*

11.1.1.7 Driver

The following script creates the Driver table.

```
CREATE TABLE
  IF NOT EXISTS Driver (
    License_number VC NOT NULL,
    Driving_experience_years INT NOT NULL,
    License_expiry_date DATE NOT NULL,
    Employee_SSN VC NOT NULL,
    CONSTRAINT PK_DRIVER PRIMARY KEY (License_number),
    CONSTRAINT Driver_FK_Employee_SSN FOREIGN KEY (Employee_SSN) REFERENCES
        Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Driver_CK_Driving_experience_years CHECK (
        Driving_experience_years > 0),
    CONSTRAINT Driver_CK_License_expiry_date CHECK (License_expiry_date >
        CURRENT_DATE)
  );
```

Code Snippet 11.8 : *Create Driver Table*

11.1.1.8 Customer

The following script creates the Customer table.

```
CREATE TABLE
  IF NOT EXISTS Customer (
    Phone_number VC NOT NULL,
    Email VC NOT NULL,
    First_name VC NOT NULL,
    Last_name VC NOT NULL,
    Gender VC,
    Registration_date DATE NOT NULL,
    Password_hashed VC NOT NULL,
    Date_of_birth DATE NOT NULL,
    Country VC NOT NULL,
    State VC NOT NULL,
    City VC NOT NULL,
    Street VC NOT NULL,
    Building INT NOT NULL,
    Apartment INT NOT NULL,
    CONSTRAINT PK_Customer PRIMARY KEY (Phone_number),
    CONSTRAINT Customer_CK_Gender CHECK (Gender IN ('Male', 'Female')),
    CONSTRAINT Customer_CK_Email CHECK (Email LIKE '%@%.%'),
    CONSTRAINT Customer_CK_Password CHECK (LENGTH (Password_hashed) >= 8),
    CONSTRAINT Customer_CK_Date_of_birth CHECK (Date_of_birth <
        CURRENT_DATE),
    CONSTRAINT Customer_CK_Registration_date CHECK (Registration_date <
        CURRENT_DATE),
    CONSTRAINT Customer_CK_Apartment CHECK (Apartment >= 0),
    CONSTRAINT Customer_CK_Building CHECK (Building >= 0),
    CONSTRAINT Customer_UK_Email UNIQUE (Email),
    CONSTRAINT Customer_UK_Phone_number UNIQUE (Phone_number),
    CONSTRAINT Customer_UK_LOCATION UNIQUE (Country, State, City, Street,
        Building, Apartment)
  );
```

Code Snippet 11.9 : *Create Customer Table*

11.1.1.9 Order

The following script creates the Order table.

```
CREATE TABLE
  IF NOT EXISTS Orders (
    Order_id ID NOT NULL,
    Notes TEXT,
    Payment_method VC NOT NULL,
    Total_amount INT NOT NULL,
    Is_online BOOLEAN NOT NULL DEFAULT FALSE,
    Employee_SSN VC,
```

```
    Customer_phone_number PHONE_NUMBER NOT NULL,
    Date DATE NOT NULL,
    Driver_license_number VC,
    CONSTRAINT PK_Order PRIMARY KEY (Order_id),
    CONSTRAINT Order_FK_Employee_SSN FOREIGN KEY (Employee_SSN) REFERENCES
        Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Order_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Order_FK_Driver_license_number FOREIGN KEY (
        Driver_license_number) REFERENCES Driver (License_number) ON UPDATE
        CASCADE ON DELETE SET NULL,
    CONSTRAINT Order_CK_Is_online CHECK (Is_online IN ('0', '1')),
    CONSTRAINT Order_CK_Total_amount CHECK (Total_amount > 0),
    CONSTRAINT Order_CK_Date CHECK (Date < CURRENT_DATE),
    CONSTRAINT Order_CK_Payment_method CHECK (Payment_method IN ('Cash', '
        Credit Card', 'Debit Card', 'PayPal', 'Apple Pay', 'Google Pay')),
    CONSTRAINT Order_UK_Driver_license_number UNIQUE (Driver_license_number
        )
  );
```

Code Snippet 11.10 : *Create Order Table*

11.1.1.10 Purchased

The following script creates the Purchased table.

```
CREATE TABLE
  IF NOT EXISTS Purchased (
    Product_SKU VC NOT NULL,
    Order_id ID NOT NULL,
    Quantity INT NOT NULL,
    Amount DECIMAL NOT NULL,
    CONSTRAINT PK_Purchased PRIMARY KEY (Product_SKU, Order_id),
    CONSTRAINT Purchased_FK_Product_SKU FOREIGN KEY (Product_SKU)
        REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Purchased_FK_Order_id FOREIGN KEY (Order_id) REFERENCES
        Orders (Order_id) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Purchased_CK_Amount CHECK (Amount >= 0),
    CONSTRAINT Purchased_CK_Quantity CHECK (Quantity >= 0)
  );
```

Code Snippet 11.11 : *Create Purchased Table*

11.1.1.11 Reviews

The following script creates the Reviews table.

```
CREATE TABLE
```

```
  IF NOT EXISTS Reviews (
    Product_SKU VC NOT NULL,
    Customer_phone_number PHONE_NUMBER NOT NULL,
    Review_date DATE NOT NULL,
    Rating INT NOT NULL,
    Comment TEXT,
    Description TEXT,
    CONSTRAINT PK_Reviews PRIMARY KEY (Product_SKU, Customer_phone_number),
    CONSTRAINT Reviews_FK_Product_SKU FOREIGN KEY (Product_SKU) REFERENCES
        Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Reviews_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Reviews_CK_Rating CHECK (
      Rating >= 1
      AND Rating <= 5
    ),
    CONSTRAINT Reviews_CK_Comment CHECK (Review_date <= CURRENT_DATE)
  );
```

Code Snippet 11.12 : *Create Reviews Table*

11.1.1.12 Support Ticket

The following script creates the Support Ticket table.

```
CREATE TABLE
  IF NOT EXISTS Support_Ticket (
    Ticket_id ID NOT NULL,
    Description TEXT NOT NULL,
    Subject TEXT NOT NULL,
    Status VARCHAR(8) NOT NULL,
    Priority VARCHAR(6) NOT NULL,
    Employee_SSN VC NOT NULL,
    Customer_phone_number VC NOT NULL,
    CONSTRAINT PK_Support_Ticket PRIMARY KEY (Ticket_id),
    CONSTRAINT Support_Ticket_CK_STATUS CHECK (Status IN ('Open', 'Closed',
        'Resolved')),
    CONSTRAINT Support_Ticket_CK_PRIORITY CHECK (Priority IN ('High', '
        Medium', 'Low')),
    CONSTRAINT Support_Ticket_FK_Employee_SSN FOREIGN KEY (Employee_SSN)
        REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Support_Ticket_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE
  );
```

Code Snippet 11.13 : *Create Support Ticket Table*

11.1.1.13 Wishlist

The following script creates the Wishlist table.

```
CREATE TABLE
  IF NOT EXISTS Wishlist (
    Product_SKU VC NOT NULL,
    Customer_phone_number PHONE_NUMBER NOT NULL,
    Total_amount REAL NOT NULL,
    CONSTRAINT PK_Wishlist PRIMARY KEY (Product_SKU, Customer_phone_number)
        ,
    CONSTRAINT Wishlist_FK_Product_SKU FOREIGN KEY (Product_SKU) REFERENCES
         Product (SKU) ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Wishlist_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Wishlist_CK_Total_amount CHECK (Total_amount >= 0)
  );
```

Code Snippet 11.14 : *Create Wishlist Table*

11.1.1.14 Working hours

The following script creates the Working hours table.

```
CREATE TABLE
  IF NOT EXISTS Working_Hours (
    Branch_phone_number PHONE_NUMBER NOT NULL,
    Day VC NOT NULL,
    Opening_hour TIME,
    Closing_hour TIME,
    CONSTRAINT PK_Working_hours PRIMARY KEY (Branch_phone_number, Day),
    CONSTRAINT Working_hours_FK_Branch_phone_number FOREIGN KEY (
        Branch_phone_number) REFERENCES Branch (phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Working_hours_CK_Day CHECK (Day IN ('Monday', 'Tuesday', '
        Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')),
    CONSTRAINT Working_hours_CK_Opening_Closing_hour CHECK (
      (
        Opening_hour IS NOT NULL
        AND Closing_hour IS NOT NULL
      )
      OR (
        Opening_hour IS NULL
        AND Closing_hour IS NULL
      )
    )
  );
```

Code Snippet 11.15 : *Create Working Hours Table*

11.1.1.15 Dependent

The following script creates the Dependent table.

```
CREATE TABLE
  IF NOT EXISTS Dependent (
    Employee_SSN VC NOT NULL,
    Name VC NOT NULL,
    Gender VARCHAR(6) NOT NULL,
    Date_of_birth DATE NOT NULL,
    Relationship VC NOT NULL,
    CONSTRAINT PK_DEPENDENT PRIMARY KEY (Employee_SSN, Name),
    CONSTRAINT Dependent_FK_Employee_SSN FOREIGN KEY (Employee_SSN)
        REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Dependent_CK_Gender CHECK (Gender IN ('Male', 'Female')),
    CONSTRAINT Dependent_CK_Date_of_birth CHECK (Date_of_birth <
        CURRENT_DATE)
  );
```

Code Snippet 11.16 : *DCreate ependent Table*

11.1.1.16 Product Image URLs

The following script creates the Product Image URLs table.

```
CREATE TABLE
  IF NOT EXISTS Product_Image_URLs (
    Product_SKU VC NOT NULL,
    Product_Image_URL VC NOT NULL,
    CONSTRAINT PK_Product_IMAGE_URLS PRIMARY KEY (Product_SKU,
        Product_Image_URL),
    CONSTRAINT Product_Image_URLs_FK_Product_Sku FOREIGN KEY (Product_SKU)
        REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE
  );
```

Code Snippet 11.17 : *Create Product Image URLs Table*

11.1.1.17 Review Image URLs

The following script creates the Review Image URLs table.

```
CREATE TABLE
  IF NOT EXISTS Review_Image_URLs (
    Product_SKU VC NOT NULL,
    Customer_phone_number PHONE_NUMBER NOT NULL,
    Product_Image_URL VC NOT NULL,
    CONSTRAINT PK_Review_IMAGE_URLS PRIMARY KEY (Product_SKU,
        Product_Image_URL),
    CONSTRAINT Review_Image_URLs_FK_Product_Sku FOREIGN KEY (Product_SKU)
        REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
```

```
    CONSTRAINT Review_Image_URLs_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Review_Image_URLs_UK_CUSTOMER_PHONE_NUMBER UNIQUE (
        Customer_phone_number)
  );
```

Code Snippet 11.18 : *Create Review Image URLs Table*

11.1.1.18 Located in

The following script creates the Located in table.

```
CREATE TABLE
  IF NOT EXISTS Located_in (
    Product_SKU VC NOT NULL,
    Branch_phone_number PHONE_NUMBER NOT NULL,
    Quantity INT NOT NULL,
    Shelf_location VC NOT NULL,
    CONSTRAINT PK_LOCATED_IN PRIMARY KEY (Product_SKU, Branch_phone_number)
        ,
    CONSTRAINT Located_in_FK_Product_SKU FOREIGN KEY (Product_SKU)
        REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Located_in_FK_Branch_phone_number FOREIGN KEY (
        Branch_phone_number) REFERENCES Branch (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
    CONSTRAINT Located_in_CK_Quantity CHECK (Quantity > 0)
  );
```

Code Snippet 11.19 : *Create Located In Table*

11.1.1.19 Colors

The following script creates the Colors table.

```
CREATE TABLE
  IF NOT EXISTS Colors (
    Product_SKU VC NOT NULL,
    Product_color VC NOT NULL,
    CONSTRAINT PK_Colors PRIMARY KEY (Product_SKU, Product_color),
    CONSTRAINT Colors_FK_Product FOREIGN KEY (Product_SKU) REFERENCES
        Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE
  );
```

Code Snippet 11.20 : *Create Colors Table*

11.1.1.20 Coupon

The following script creates the Coupon table.

```sql
CREATE TABLE
  IF NOT EXISTS Coupon (
    Code ID NOT NULL,
    Description TEXT NOT NULL,
    Discount_percent REAL NOT NULL,
    Times_used INT NOT NULL,
    Minimum_order_amount REAL NOT NULL,
    Maximum_order_amount REAL NOT NULL,
    Usage_limit INT NOT NULL,
    Valid_from DATE NOT NULL,
    Valid_to DATE NOT NULL,
    Order_ID ID NOT NULL,
    Discount_amount REAL NOT NULL,
    Redeem_date DATE NOT NULL,
    CONSTRAINT PK_Coupon PRIMARY KEY (Code),
    CONSTRAINT Coupon_FK_Order FOREIGN KEY (Order_ID) REFERENCES Orders (
        Order_id) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Coupon_CK_Percent CHECK (
      Discount_percent > 0
      AND Discount_percent <= 100
    ),
    CONSTRAINT Coupon_CK_Limit CHECK (Usage_limit > 0),
    CONSTRAINT Coupon_CK_Valid CHECK (Valid_from <= Valid_to),
    CONSTRAINT Coupon_CK_Amount CHECK (Discount_amount >= 0),
    CONSTRAINT Coupon_CK_Times_used CHECK (Times_used >= 0),
    CONSTRAINT Coupon_CK_Order_amount CHECK (
      Minimum_order_amount > 0
      AND Maximum_order_amount > 0
      AND Minimum_order_amount <= Maximum_order_amount
    ),
    CONSTRAINT Coupon_CK_Redeem_date CHECK (Redeem_date <= CURRENT_DATE),
    CONSTRAINT Coupon_UK_ORDER UNIQUE (Order_ID)
  );
```

Code Snippet 11.21 : *Create Coupon Table*

11.1.1.21 Department location

The following script creates the Department location table.

```sql
CREATE TABLE
  IF NOT EXISTS Department_Location (
    Department_name VC NOT NULL,
    Location VC NOT NULL,
    CONSTRAINT PK_Department_Location PRIMARY KEY (Department_name,
        Location),
    CONSTRAINT Department_Location_FK_Department FOREIGN KEY (
        Department_name) REFERENCES Department (Name) ON UPDATE CASCADE ON
```

```
        DELETE CASCADE
);
```

Code Snippet 11.22 : *Create Department Location Table*

11.1.2 Alter Queries

11.1.2.1 Branch

The following script alters the Branch table.

```
ALTER TABLE Branch ADD CONSTRAINT FK_Employee FOREIGN KEY (Employee_SSN)
    REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL;
```

Code Snippet 11.23 : *Alter Branch Table*

11.1.2.2 Department

The following script alters the Department table.

```
ALTER TABLE Department ADD CONSTRAINT FK_Employee FOREIGN KEY (Employee_SSN
    ) REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL;
```

Code Snippet 11.24 : *Alter Department Table*

11.1.3 Insert Queries

11.1.3.1 Supplier

The following script inserts data into the Supplier table.

```
INSERT INTO
  Supplier (Website, Supplier_name, Contact_person_email,
     Contact_person_first_name, Contact_person_last_name,
     Contact_person_phone_number)
VALUES
  ('www.techbrand.com', 'TechBrand', 'contact@techbrand.com', 'Alice', 'Doe
     ', '+96134567890'),
  ('www.computex.com', 'ComputeX', 'support@computex.com', 'John', 'Smith',
      '+96198765432'),
  ('www.comfortco.com', 'ComfortCo', 'info@comfortco.com', 'Emma', 'Johnson
     ', '+96145678901'),
  ('www.sportwear.com', 'SportWear', 'sales@sportwear.com', 'Michael', '
     Brown', '+96167890123'),
  ('www.musicpro.com', 'MusicPro', 'music@musicpro.com', 'Sarah', 'Lee', '
     +96178901234'),
  ('www.homeappl.com', 'HomeAppl', 'appliances@homeappl.com', 'David', '
     Evans', '+96189012345'),
  ('www.visionco.com', 'VisionCo', 'vision@visionco.com', 'Jessica', '
     Taylor', '+96190123456'),
```

```
  ('www.kitchenx.com', 'KitchenX', 'service@kitchenx.com', 'Kevin', 'Wilson
     ', '+96101234567'),
  ('www.fashionhub.com', 'FashionHub', 'hello@fashionhub.com', 'Emily', '
     Harris', '+96123456789'),
  ('www.timekeep.com', 'TimeKeep', 'contact@timekeep.com', 'Daniel', '
     Martin', '+96156789012'),
  ('www.edubooks.com', 'EduBooks', 'edu@edubooks.com', 'Olivia', 'White', '
     +96167890124'),
  ('www.lightpro.com', 'LightPro', 'light@lightpro.com', 'Robert', 'Scott',
      '+96178901235'),
  ('www.audiomax.com', 'AudioMax', 'audio@audiomax.com', 'Sophia', '
     Anderson', '+96189012346'),
  ('www.gamezone.com', 'GameZone', 'games@gamezone.com', 'Liam', 'Thomas',
     '+96190123457'),
  ('www.sportwearlb.com', 'SportWearLebanon', 'store@sportwear.com', 'Noah'
     , 'Miller', '+96101234568');
```

Code Snippet 11.25 : *Insert into Supplier Table*

11.1.3.2 Category

The following script inserts data into the Category table.

```
INSERT INTO
  Category (Name, Description, Parent_Category_Name)
VALUES
  ('Electronics', 'Devices and gadgets', NULL),
  ('Clothing', 'Apparel and fashion items', NULL),
  ('Furniture', 'Home and office furniture', NULL),
  ('Fashion', 'Clothing and accessories', NULL),
  ('Books', 'Printed and digital books', 'Stationery'),
  ('Groceries', 'Food and daily supplies', 'Health'),
  ('Sports', 'Sporting equipment and apparel', 'Clothing'),
  ('Beauty', 'Cosmetics and skincare products', 'Health'),
  ('Toys', 'Toys for kids and adults', 'Sports'),
  ('Automotive', 'Car parts and accessories', 'Electronics'),
  ('Jewelry', 'Watches, rings, and necklaces', 'Fashion'),
  ('Health', 'Medical supplies and equipment', 'Beauty'),
  ('Stationery', 'Office and school supplies', 'Furniture'),
  ('Pets', 'Pet food and accessories', 'Groceries'),
  ('Music', 'Instruments and music equipment', 'Art'),
  ('Art', 'Art supplies and crafts', 'Stationery');
```

Code Snippet 11.26 : *Insert into Category Table*

11.1.3.3 Product

The following script inserts data into the Product table.

```sql
INSERT INTO
  Product (SKU, Name, Price, Description, Weight, Brand, Width, Height,
     Length, Category_name, Supplier_website)
VALUES
  ('1001', 'Smartphone', 600, '5G-enabled phone', 0.2, 'TechBrand', 7, 15,
     0.8, 'Electronics', 'www.techbrand.com'),
  ('1002', 'Laptop', 1200, 'Ultrabook with 16GB RAM', 1.5, 'ComputeX', 32,
     22, 1.5, 'Electronics', 'www.computex.com'),
  ('1003', 'Office Chair', 150, 'Ergonomic chair', 12, 'ComfortCo', 60,
     120, 60, 'Furniture', 'www.comfortco.com'),
  ('1004', 'Running Shoes', 100, 'Lightweight shoes', 0.5, 'SportWear', 12,
      35, 10, 'Sports', 'www.sportwear.com'),
  ('1005', 'Acoustic Guitar', 300, '6-string guitar', 3, 'MusicPro', 38,
     100, 12, 'Music', 'www.musicpro.com'),
  ('1006', 'Refrigerator', 800, 'Double door fridge', 65, 'HomeAppl', 90,
     180, 75, 'Electronics', 'www.homeappl.com'),
  ('1007', 'LED TV', 400, '50-inch 4K UHD', 8, 'VisionCo', 112, 65, 5, '
     Electronics', 'www.visionco.com'),
  ('1008', 'Blender', 70, 'High-speed blender', 2, 'KitchenX', 20, 40, 15,
     'Electronics', 'www.kitchenx.com'),
  ('1009', 'T-Shirt', 25, 'Cotton T-shirt', 0.25, 'FashionHub', 30, 80, 1,
     'Clothing', 'www.fashionhub.com'),
  ('1010', 'Watch', 500, 'Smartwatch with GPS', 0.2, 'TimeKeep', 5, 5, 1, '
     Jewelry', 'www.timekeep.com'),
  ('1011', 'Textbook', 50, 'Advanced mathematics book', 1, 'EduBooks', 21,
     28, 3, 'Books', 'www.edubooks.com'),
  ('1012', 'Desk Lamp', 30, 'LED desk lamp', 1.5, 'LightPro', 15, 40, 15, '
     Furniture', 'www.lightpro.com'),
  ('1013', 'Wireless Headphones', 150, 'Noise-canceling headphones', 0.3, '
     AudioMax', 18, 20, 15, 'Electronics', 'www.audiomax.com'),
  ('1014', 'Soccer Ball', 40, 'FIFA approved', 0.45, 'SportWear', 20, 20,
     20, 'Sports', 'www.sportwear.com'),
  ('1015', 'Gaming Console', 500, 'Next-gen console', 3, 'GameZone', 30,
     10, 25, 'Electronics', 'www.gamezone.com');
```

Code Snippet 11.27 : *Insert into Product Table*

11.1.3.4 Branch & Department & Employee

Step 1: Insert departments with Employee_SSN as NULL

```sql
INSERT INTO
  Department (Name, Number_of_employees, Employee_SSN,
     Manager_start_date)
VALUES
  ('Sales', 6, NULL, '2022-03-01'),
  ('Marketing', 2, NULL, '2021-06-15'),
  ('HR', 2, NULL, '2023-01-10'),
```

```
  ('Finance', 3, NULL, '2019-10-05'),
  ('Operations', 2, NULL, '2020-08-25'),
  ('IT', 1, NULL, '2024-04-18'),
  ('Customer Support', 2, NULL, '2022-09-12'),
  ('Logistics', 3, NULL, '2020-11-20'),
  ('Legal', 1, NULL, '2021-02-14'),
  ('Training', 4, NULL, '2023-03-27'),
  ('Facilities', 1, NULL, '2019-06-01'),
  ('R&D', 1, NULL, '2022-05-10');
```

Code Snippet 11.28 : *Insert into Department Table*

Step 2: Insert branches with Employee_SSN as NULL

```
INSERT INTO
  Branch (Phone_number, Name, Country, State, City, Street, Building,
     Apartment, Employee_SSN)
VALUES
  ('+96111111111', 'Beirut Main', 'Lebanon', 'Beirut', 'Beirut', '
     Hamra', 10, 12, NULL),
  ('+96122222222', 'Tripoli Branch', 'Lebanon', 'North', 'Tripoli', '
     Mina Street', 5, 12, NULL),
  ('+96133333333', 'Sidon Hub', 'Lebanon', 'South', 'Sidon', 'Corniche
     ', 3, 6, NULL),
  ('+96144444444', 'Zahle Branch', 'Lebanon', 'Beqaa', 'Zahle', 'Beka
     St', 6, 4, NULL),
  ('+96155555555', 'Jounieh Store', 'Lebanon', 'Mount Lebanon', '
     Jounieh', 'Coastal Rd', 5, 3, NULL),
  ('+96166666666', 'Byblos Outlet', 'Lebanon', 'Mount Lebanon', '
     Byblos', 'Roman Street', 6, 7, NULL),
  ('+96177777777', 'Tyre Shop', 'Lebanon', 'South', 'Tyre', 'Port Road
     ', 2, 1, NULL),
  ('+96188888888', 'Baalbek Point', 'Lebanon', 'Beqaa', 'Baalbek', '
     Temple Rd', 3, 2, NULL),
  ('+96199999999', 'Batroun Corner', 'Lebanon', 'North', 'Batroun', '
     Old City', 1, 1, NULL),
  ('+96112345678', 'Downtown Center', 'Lebanon', 'Beirut', 'Beirut', '
     Downtown', 9, 11, NULL),
  ('+96124681357', 'Dora Warehouse', 'Lebanon', 'Mount Lebanon', 'Dora
     ', 'Industrial Zone', 3, 10, NULL),
  ('+96165432198', 'Aley Branch', 'Lebanon', 'Mount Lebanon', 'Aley',
     'Souk Street', 6, 5, NULL),
  ('+96111223344', 'Choueifat Station', 'Lebanon', 'Mount Lebanon', '
     Choueifat', 'Railway Rd', 7, 3, NULL);
```

Code Snippet 11.29 : *Insert into Branch Table*

Step 3: Insert employees referencing Department_name

```sql
INSERT INTO
  Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth,
     Email, First_name, Last_name, Phone_number, Country, State, City,
      Street, Building, Apartment, Branch_phone_number, Supervisor_SSN
     , Department_name)
VALUES
  -- Sales Department Employees
  (123456789, 'Manager', 100000, '2019-01-15', 'Male', '1980-05-20', '
     john.doe@example.com', 'John', 'Doe', '+96100000001', 'Lebanon',
     'Beirut', 'Beirut', 'Hamra', 10, 12, '+96111111111', NULL, 'Sales
     '),
  (111111111, 'AdminAssistant', 50000, '2020-02-20', 'Female', '
     1990-08-15', 'jane.smith@example.com', 'Jane', 'Smith', '
     +96100000002', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 13, '
     +96111111111', 123456789, 'Sales'),
  (111111112, 'Trainer', 60000, '2021-05-10', 'Male', '1985-11-30', '
     peter.brown@example.com', 'Peter', 'Brown', '+96100000003', '
     Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 14, '+96111111111',
     123456789, 'Sales'),
  (111111113, 'ITSpecialist', 70000, '2021-07-01', 'Male', '1992-03-12
     ', 'alex.jones@example.com', 'Alex', 'Jones', '+96100000004', '
     Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 15, '+96111111111',
     123456789, 'Sales'),
  (111111114, 'ProcurementOfficer', 55000, '2022-09-15', 'Female', '
     1995-06-20', 'lisa.green@example.com', 'Lisa', 'Green', '
     +96100000005', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 16, '
     +96111111111', 123456789, 'Sales'),
  (654321987, 'AdminAssistant', 52000, '2021-09-10', 'Male', '
     1991-06-14', 'steven.king@example.com', 'Steven', 'King', '
     +96100000028', 'Lebanon', 'Mount Lebanon', 'Aley', 'Souk Street',
      6, 5, '+96165432198', 123456789, 'Sales'),
  -- Marketing Department Employees
  (987654321, 'MarketingHead', 90000, '2018-06-10', 'Female', '
     1982-09-25', 'sarah.johnson@example.com', 'Sarah', 'Johnson', '
     +96100000006', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5,
     12, '+96122222222', NULL, 'Marketing'),
  (222222222, 'AdminAssistant', 50000, '2021-08-01', 'Male', '
     1990-12-12', 'michael.taylor@example.com', 'Michael', 'Taylor', '
     +96100000007', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5,
     13, '+96122222222', 987654321, 'Marketing'),
  -- HR Department Employees
  (123459876, 'HRManager', 85000, '2023-01-10', 'Male', '1984-03-15',
     'david.wilson@example.com', 'David', 'Wilson', '+96100000008', '
     Lebanon', 'Beqaa', 'Zahle', 'Beka St', 6, 4, '+96144444444', NULL
     , 'HR'),
  (333333333, 'AdminAssistant', 48000, '2023-02-15', 'Female', '
     1991-07-22', 'emily.davis@example.com', 'Emily', 'Davis', '
```

```
    +96100000009', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St', 6, 5, '
    +96144444444', 123459876, 'HR'),
-- Finance Department Employees
(567894321, 'FinanceManager', 95000, '2019-10-05', 'Female', '
    1978-11-08', 'laura.martin@example.com', 'Laura', 'Martin', '
    +96100000010', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
    ', 5, 3, '+96155555555', NULL, 'Finance'),
(444444444, 'AdminAssistant', 47000, '2020-03-12', 'Male', '
    1989-04-18', 'daniel.moore@example.com', 'Daniel', 'Moore', '
    +96100000011', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
    ', 5, 4, '+96155555555', 567894321, 'Finance'),
(555555555, 'ComplianceOfficer', 62000, '2021-07-23', 'Female', '
    1993-09-30', 'olivia.thomas@example.com', 'Olivia', 'Thomas', '
    +96100000012', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
    ', 5, 5, '+96155555555', 567894321, 'Finance'),
-- Operations Department Employees
(543216789, 'OperationsHead', 90000, '2018-08-15', 'Male', '
    1980-07-22', 'mark.stevens@example.com', 'Mark', 'Stevens', '
    +96100000013', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 6, '
    +96133333333', NULL, 'Operations'),
(666666666, 'AdminAssistant', 50000, '2019-05-20', 'Female', '
    1992-01-15', 'anna.miller@example.com', 'Anna', 'Miller', '
    +96100000014', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 7, '
    +96133333333', 543216789, 'Operations'),
-- IT Department Employee
(678912345, 'ITSpecialist', 85000, '2020-03-10', 'Male', '1983-12-05
    ', 'kevin.lee@example.com', 'Kevin', 'Lee', '+96100000015', '
    Lebanon', 'Mount Lebanon', 'Byblos', 'Roman Street', 6, 7, '
    +96166666666', NULL, 'IT'),
-- Customer Support Department Employees
(876543219, 'SupportManager', 80000, '2017-11-01', 'Female', '
    1985-04-18', 'laura.kim@example.com', 'Laura', 'Kim', '
    +96100000016', 'Lebanon', 'South', 'Tyre', 'Port Road', 2, 1, '
    +96177777777', NULL, 'Customer Support'),
(777777777, 'AdminAssistant', 48000, '2019-06-15', 'Male', '
    1990-08-30', 'john.park@example.com', 'John', 'Park', '
    +96100000017', 'Lebanon', 'South', 'Tyre', 'Port Road', 2, 2, '
    +96177777777', 876543219, 'Customer Support'),
-- Logistics Department Employees
(345678912, 'LogisticsHead', 90000, '2015-04-01', 'Male', '
    1979-02-25', 'peter.johnson@example.com', 'Peter', 'Johnson', '
    +96100000018', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 2,
    '+96188888888', NULL, 'Logistics'),
(888888881, 'AdminAssistant', 47000, '2018-05-10', 'Female', '
    1991-09-15', 'susan.martin@example.com', 'Susan', 'Martin', '
    +96100000019', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 3,
    '+96188888888', 345678912, 'Logistics'),
(888888882, 'ProcurementOfficer', 55000, '2019-07-22', 'Male', '
```

```
    1988-12-08', 'brian.davis@example.com', 'Brian', 'Davis', '
    +96100000020', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 4,
    '+96188888888', 345678912, 'Logistics'),
  -- Legal Department Employee
  (456789123, 'LegalAdvisor', 95000, '2016-09-05', 'Male', '1977-11-11
    ', 'richard.harris@example.com', 'Richard', 'Harris', '
    +96100000021', 'Lebanon', 'North', 'Batroun', 'Old City', 1, 1, '
    +96199999999', NULL, 'Legal'),
  -- Training Department Employees
  (789123456, 'Trainer', 80000, '2020-01-05', 'Female', '1984-05-25',
    'emily.clark@example.com', 'Emily', 'Clark', '+96100000022', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 10, '
    +96124681357', NULL, 'Training'),
  (999999991, 'Trainer', 60000, '2021-04-12', 'Male', '1992-03-18', '
    michael.brown@example.com', 'Michael', 'Brown', '+96100000023', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 11, '
    +96124681357', 789123456, 'Training'),
  (999999992, 'Trainer', 61000, '2021-07-30', 'Female', '1989-09-07',
    'jessica.wilson@example.com', 'Jessica', 'Wilson', '+96100000024'
    , 'Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 12, '
    +96124681357', 789123456, 'Training'),
  (999999993, 'AdminAssistant', 50000, '2022-02-18', 'Male', '
    1994-12-22', 'robert.moore@example.com', 'Robert', 'Moore', '
    +96100000025', 'Lebanon', 'Mount Lebanon', 'Dora', 'Industrial
    Zone', 3, 13, '+96124681357', 789123456, 'Training'),
  -- Facilities Department Employee
  (987123456, 'FacilitiesManager', 85000, '2016-03-15', 'Male', '
    1975-08-05', 'george.anderson@example.com', 'George', 'Anderson',
     '+96100000026', 'Lebanon', 'Mount Lebanon', 'Choueifat', '
    Railway Rd', 7, 3, '+96111223344', NULL, 'Facilities'),
  -- R&D Department Employee
  (234567891, 'R&DManager', 95000, '2020-05-10', 'Female', '1983-02-20
    ', 'rachel.adams@example.com', 'Rachel', 'Adams', '+96100000027',
     'Lebanon', 'Beirut', 'Beirut', 'Downtown', 9, 11, '+96112345678'
    , NULL, 'R&D');
```

Code Snippet 11.30 : *Insert into Employee Table*

Step 4: Update departments to set Employee_SSN (manager's SSN)

```
UPDATE Department
SET
  Employee_SSN = CASE Name
    WHEN 'Sales' THEN '123456789'
    WHEN 'Marketing' THEN '987654321'
    WHEN 'HR' THEN '123459876'
    WHEN 'Finance' THEN '567894321'
    WHEN 'Operations' THEN '543216789'
    WHEN 'IT' THEN '678912345'
```

```
    WHEN 'Customer Support' THEN '876543219'
    WHEN 'Logistics' THEN '345678912'
    WHEN 'Legal' THEN '456789123'
    WHEN 'Training' THEN '789123456'
    WHEN 'Facilities' THEN '987123456'
    WHEN 'R&D' THEN '234567891'
    ELSE Employee_SSN
  END;
```

Code Snippet 11.31 : *Update Department Table*

Step 5: Update branches to set Employee_SSN

```
UPDATE Branch
SET
  Employee_SSN = CASE Phone_number
    WHEN '+96111111111' THEN '123456789'
    WHEN '+96122222222' THEN '987654321'
    WHEN '+96133333333' THEN '543216789'
    WHEN '+96144444444' THEN '123459876'
    WHEN '+96155555555' THEN '567894321'
    WHEN '+96166666666' THEN '678912345'
    WHEN '+96177777777' THEN '876543219'
    WHEN '+96188888888' THEN '345678912'
    WHEN '+96199999999' THEN '456789123'
    WHEN '+96112345678' THEN '234567891'
    WHEN '+96124681357' THEN '789123456'
    WHEN '+96165432198' THEN '654321987'
    WHEN '+96111223344' THEN '987123456'
    ELSE Employee_SSN
  END;
```

Code Snippet 11.32 : *Update Branch Table*

Finally wrap the whole code with a transaction block. Add `BEGIN;` at the beginning and `COMMIT;` at the end.

The final script is as follows:

```
BEGIN;

-- Step 1: Insert departments with Employee_SSN as NULL
INSERT INTO
  Department (Name, Number_of_employees, Employee_SSN, Manager_start_date)
VALUES
  ('Sales', 6, NULL, '2022-03-01'),
  ('Marketing', 2, NULL, '2021-06-15'),
  ('HR', 2, NULL, '2023-01-10'),
  ('Finance', 3, NULL, '2019-10-05'),
  ('Operations', 2, NULL, '2020-08-25'),
```

```
  ('IT', 1, NULL, '2024-04-18'),
  ('Customer Support', 2, NULL, '2022-09-12'),
  ('Logistics', 3, NULL, '2020-11-20'),
  ('Legal', 1, NULL, '2021-02-14'),
  ('Training', 4, NULL, '2023-03-27'),
  ('Facilities', 1, NULL, '2019-06-01'),
  ('R&D', 1, NULL, '2022-05-10');

-- Step 2: Insert branches with Employee_SSN as NULL
INSERT INTO
  Branch (Phone_number, Name, Country, State, City, Street, Building,
      Apartment, Employee_SSN)
VALUES
  ('+96111111111', 'Beirut Main', 'Lebanon', 'Beirut', 'Beirut', 'Hamra',
      10, 12, NULL),
  ('+96122222222', 'Tripoli Branch', 'Lebanon', 'North', 'Tripoli', 'Mina
      Street', 5, 12, NULL),
  ('+96133333333', 'Sidon Hub', 'Lebanon', 'South', 'Sidon', 'Corniche', 3,
       6, NULL),
  ('+96144444444', 'Zahle Branch', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St',
      6, 4, NULL),
  ('+96155555555', 'Jounieh Store', 'Lebanon', 'Mount Lebanon', 'Jounieh',
      'Coastal Rd', 5, 3, NULL),
  ('+96166666666', 'Byblos Outlet', 'Lebanon', 'Mount Lebanon', 'Byblos', '
      Roman Street', 6, 7, NULL),
  ('+96177777777', 'Tyre Shop', 'Lebanon', 'South', 'Tyre', 'Port Road', 2,
       1, NULL),
  ('+96188888888', 'Baalbek Point', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple
      Rd', 3, 2, NULL),
  ('+96199999999', 'Batroun Corner', 'Lebanon', 'North', 'Batroun', 'Old
      City', 1, 1, NULL),
  ('+96112345678', 'Downtown Center', 'Lebanon', 'Beirut', 'Beirut', '
      Downtown', 9, 11, NULL),
  ('+96124681357', 'Dora Warehouse', 'Lebanon', 'Mount Lebanon', 'Dora', '
      Industrial Zone', 3, 10, NULL),
  ('+96165432198', 'Aley Branch', 'Lebanon', 'Mount Lebanon', 'Aley', 'Souk
       Street', 6, 5, NULL),
  ('+96111223344', 'Choueifat Station', 'Lebanon', 'Mount Lebanon', '
      Choueifat', 'Railway Rd', 7, 3, NULL);

-- Step 3: Insert employees referencing Department_name
INSERT INTO
  Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email,
       First_name, Last_name, Phone_number, Country, State, City, Street,
      Building, Apartment, Branch_phone_number, Supervisor_SSN,
      Department_name)
VALUES
  -- Sales Department Employees
```

```
('123456789', 'Manager', 100000, '2019-01-15', 'Male', '1980-05-20', '
   john.doe@example.com', 'John', 'Doe', '+96100000001', 'Lebanon', '
   Beirut', 'Beirut', 'Hamra', 10, 12, '+96111111111', NULL, 'Sales'),
('111111111', 'AdminAssistant', 50000, '2020-02-20', 'Female', '
   1990-08-15', 'jane.smith@example.com', 'Jane', 'Smith', '+96100000002'
   , 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 13, '+96111111111',
   123456789, 'Sales'),
('111111112', 'Trainer', 60000, '2021-05-10', 'Male', '1985-11-30', '
   peter.brown@example.com', 'Peter', 'Brown', '+96100000003', 'Lebanon',
    'Beirut', 'Beirut', 'Hamra', 10, 14, '+96111111111', 123456789, '
   Sales'),
('111111113', 'ITSpecialist', 70000, '2021-07-01', 'Male', '1992-03-12',
   'alex.jones@example.com', 'Alex', 'Jones', '+96100000004', 'Lebanon',
   'Beirut', 'Beirut', 'Hamra', 10, 15, '+96111111111', 123456789, 'Sales
   '),
('111111114', 'ProcurementOfficer', 55000, '2022-09-15', 'Female', '
   1995-06-20', 'lisa.green@example.com', 'Lisa', 'Green', '+96100000005'
   , 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 16, '+96111111111',
   123456789, 'Sales'),
('654321987', 'AdminAssistant', 52000, '2021-09-10', 'Male', '1991-06-14'
   , 'steven.king@example.com', 'Steven', 'King', '+96100000028', '
   Lebanon', 'Mount Lebanon', 'Aley', 'Souk Street', 6, 5, '+96165432198'
   , 123456789, 'Sales'),
-- Marketing Department Employees
('987654321', 'MarketingHead', 90000, '2018-06-10', 'Female', '1982-09-25
   ', 'sarah.johnson@example.com', 'Sarah', 'Johnson', '+96100000006', '
   Lebanon', 'North', 'Tripoli', 'Mina Street', 5, 12, '+96122222222',
   NULL, 'Marketing'),
('222222222', 'AdminAssistant', 50000, '2021-08-01', 'Male', '1990-12-12'
   , 'michael.taylor@example.com', 'Michael', 'Taylor', '+96100000007', '
   Lebanon', 'North', 'Tripoli', 'Mina Street', 5, 13, '+96122222222',
   987654321, 'Marketing'),
-- HR Department Employees
('123459876', 'HRManager', 85000, '2023-01-10', 'Male', '1984-03-15', '
   david.wilson@example.com', 'David', 'Wilson', '+96100000008', 'Lebanon
   ', 'Beqaa', 'Zahle', 'Beka St', 6, 4, '+96144444444', NULL, 'HR'),
('333333333', 'AdminAssistant', 48000, '2023-02-15', 'Female', '
   1991-07-22', 'emily.davis@example.com', 'Emily', 'Davis', '
   +96100000009', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St', 6, 5, '
   +96144444444', 123459876, 'HR'),
-- Finance Department Employees
('567894321', 'FinanceManager', 95000, '2019-10-05', 'Female', '
   1978-11-08', 'laura.martin@example.com', 'Laura', 'Martin', '
   +96100000010', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5,
    3, '+96155555555', NULL, 'Finance'),
('444444444', 'AdminAssistant', 47000, '2020-03-12', 'Male', '1989-04-18'
   , 'daniel.moore@example.com', 'Daniel', 'Moore', '+96100000011', '
   Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5, 4, '
```

```
    +96155555555', 567894321, 'Finance'),
('555555555', 'ComplianceOfficer', 62000, '2021-07-23', 'Female', '
    1993-09-30', 'olivia.thomas@example.com', 'Olivia', 'Thomas', '
    +96100000012', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5,
     5, '+96155555555', 567894321, 'Finance'),
-- Operations Department Employees
('543216789', 'OperationsHead', 90000, '2018-08-15', 'Male', '1980-07-22'
    , 'mark.stevens@example.com', 'Mark', 'Stevens', '+96100000013', '
    Lebanon', 'South', 'Sidon', 'Corniche', 3, 6, '+96133333333', NULL, '
    Operations'),
('666666666', 'AdminAssistant', 50000, '2019-05-20', 'Female', '
    1992-01-15', 'anna.miller@example.com', 'Anna', 'Miller', '
    +96100000014', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 7, '
    +96133333333', 543216789, 'Operations'),
-- IT Department Employee
('678912345', 'ITSpecialist', 85000, '2020-03-10', 'Male', '1983-12-05',
    'kevin.lee@example.com', 'Kevin', 'Lee', '+96100000015', 'Lebanon', '
    Mount Lebanon', 'Byblos', 'Roman Street', 6, 7, '+96166666666', NULL,
    'IT'),
-- Customer Support Department Employees
('876543219', 'SupportManager', 80000, '2017-11-01', 'Female', '
    1985-04-18', 'laura.kim@example.com', 'Laura', 'Kim', '+96100000016',
    'Lebanon', 'South', 'Tyre', 'Port Road', 2, 1, '+96177777777', NULL, '
    Customer Support'),
('777777777', 'AdminAssistant', 48000, '2019-06-15', 'Male', '1990-08-30'
    , 'john.park@example.com', 'John', 'Park', '+96100000017', 'Lebanon',
    'South', 'Tyre', 'Port Road', 2, 2, '+96177777777', 876543219, '
    Customer Support'),
-- Logistics Department Employees
('345678912', 'LogisticsHead', 90000, '2015-04-01', 'Male', '1979-02-25',
     'peter.johnson@example.com', 'Peter', 'Johnson', '+96100000018', '
    Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 2, '+96188888888', NULL,
     'Logistics'),
('888888881', 'AdminAssistant', 47000, '2018-05-10', 'Female', '
    1991-09-15', 'susan.martin@example.com', 'Susan', 'Martin', '
    +96100000019', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 3, '
    +96188888888', 345678912, 'Logistics'),
('888888882', 'ProcurementOfficer', 55000, '2019-07-22', 'Male', '
    1988-12-08', 'brian.davis@example.com', 'Brian', 'Davis', '
    +96100000020', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 4, '
    +96188888888', 345678912, 'Logistics'),
-- Legal Department Employee
('456789123', 'LegalAdvisor', 95000, '2016-09-05', 'Male', '1977-11-11',
    'richard.harris@example.com', 'Richard', 'Harris', '+96100000021', '
    Lebanon', 'North', 'Batroun', 'Old City', 1, 1, '+96199999999', NULL,
    'Legal'),
-- Training Department Employees
('789123456', 'Trainer', 80000, '2020-01-05', 'Female', '1984-05-25', '
```

```
    emily.clark@example.com', 'Emily', 'Clark', '+96100000022', 'Lebanon',
     'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 10, '+96124681357',
    NULL, 'Training'),
('999999991', 'Trainer', 60000, '2021-04-12', 'Male', '1992-03-18', '
    michael.brown@example.com', 'Michael', 'Brown', '+96100000023', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 11, '
    +96124681357', 789123456, 'Training'),
('999999992', 'Trainer', 61000, '2021-07-30', 'Female', '1989-09-07', '
    jessica.wilson@example.com', 'Jessica', 'Wilson', '+96100000024', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 12, '
    +96124681357', 789123456, 'Training'),
('999999993', 'AdminAssistant', 50000, '2022-02-18', 'Male', '1994-12-22'
    , 'robert.moore@example.com', 'Robert', 'Moore', '+96100000025', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 13, '
    +96124681357', 789123456, 'Training'),
-- Facilities Department Employee
('987123456', 'FacilitiesManager', 85000, '2016-03-15', 'Male', '
    1975-08-05', 'george.anderson@example.com', 'George', 'Anderson', '
    +96100000026', 'Lebanon', 'Mount Lebanon', 'Choueifat', 'Railway Rd',
    7, 3, '+96111223344', NULL, 'Facilities'),
-- R&D Department Employee
('234567891', 'R&DManager', 95000, '2020-05-10', 'Female', '1983-02-20',
    'rachel.adams@example.com', 'Rachel', 'Adams', '+96100000027', '
    Lebanon', 'Beirut', 'Beirut', 'Downtown', 9, 11, '+96112345678', NULL,
     'R&D'),
-- Driver Employees
('023456789', 'Driver', 35000, '2020-05-12', 'Male', '1988-03-15', '
    michael.jordan@nexstore.io', 'Michael', 'Jordan', '+96100000030', '
    Lebanon', 'Beirut', 'Beirut', 'Main Street', 5, 2, '+96111111111',
    NULL, 'Logistics'),
('087654321', 'Driver', 32000, '2021-07-18', 'Female', '1992-07-20', '
    emily.clarkson@nexstore.io', 'Emily', 'Clarkson', '+96100000031', '
    Lebanon', 'North', 'Tripoli', 'Sea Road', 3, 1, '+96122222222', NULL,
    'Logistics'),
('067894321', 'Driver', 38000, '2019-09-03', 'Male', '1985-11-11', '
    richard.lewis@nexstore.io', 'Richard', 'Lewis', '+96100000032', '
    Lebanon', 'Mount Lebanon', 'Jounieh', 'Harbor Lane', 2, 5, '
    +96155555555', NULL, 'Logistics'),
('043216789', 'Driver', 37000, '2022-01-22', 'Female', '1990-04-30', '
    anna.roberts@nexstore.io', 'Anna', 'Roberts', '+96100000033', 'Lebanon
    ', 'South', 'Sidon', 'Coastal Road', 6, 4, '+96133333333', NULL, '
    Logistics'),
('078912345', 'Driver', 34000, '2021-12-14', 'Male', '1989-08-25', 'kevin
    .baker@nexstore.io', 'Kevin', 'Baker', '+96100000034', 'Lebanon', '
    Beqaa', 'Zahle', 'Valley Drive', 1, 3, '+96144444444', NULL, '
    Logistics'),
('076543219', 'Driver', 36000, '2020-11-01', 'Female', '1987-09-18', '
    olivia.james@nexstore.io', 'Olivia', 'James', '+96100000035', 'Lebanon
```

```
    ', 'South', 'Tyre', 'Marina Rd', 2, 6, '+96177777777', NULL, '
    Logistics'),
('045678912', 'Driver', 40000, '2018-06-20', 'Male', '1982-12-06', 'john.
    ryan@nexstore.io', 'John', 'Ryan', '+96100000036', 'Lebanon', 'Beqaa',
     'Baalbek', 'Temple Avenue', 4, 2, '+96188888888', NULL, 'Logistics'),
('056789123', 'Driver', 39000, '2017-04-25', 'Female', '1984-05-15', '
    susan.richards@nexstore.io', 'Susan', 'Richards', '+96100000037', '
    Lebanon', 'North', 'Batroun', 'Market St', 3, 1, '+96199999999', NULL,
     'Logistics'),
('034567891', 'Driver', 37000, '2020-08-12', 'Male', '1988-11-27', 'mark.
    harrison@nexstore.io', 'Mark', 'Harrison', '+96100000038', 'Lebanon',
    'Beirut', 'Beirut', 'Central Rd', 5, 4, '+96112345678', NULL, '
    Logistics'),
('089123456', 'Driver', 42000, '2016-03-15', 'Female', '1980-10-10', '
    jessica.brooks@nexstore.io', 'Jessica', 'Brooks', '+96100000039', '
    Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 7, 6, '
    +96124681357', NULL, 'Logistics'),
('054321987', 'Driver', 31000, '2023-02-28', 'Male', '1995-03-05', '
    daniel.hunt@nexstore.io', 'Daniel', 'Hunt', '+96100000040', 'Lebanon',
     'Mount Lebanon', 'Aley', 'Summit Ave', 3, 5, '+96165432198', NULL, '
    Logistics'),
('021987654', 'Driver', 38000, '2018-11-30', 'Female', '1983-07-23', '
    emma.jenkins@nexstore.io', 'Emma', 'Jenkins', '+96100000041', 'Lebanon
    ', 'Mount Lebanon', 'Choueifat', 'Green Lane', 2, 3, '+96111223344',
    NULL, 'Logistics'),
('087123456', 'Driver', 40000, '2017-07-18', 'Male', '1981-06-02', 'alex.
    carter@nexstore.io', 'Alex', 'Carter', '+96100000042', 'Lebanon', '
    Mount Lebanon', 'Byblos', 'Old Harbor', 6, 7, '+96166666666', NULL, '
    Logistics'),
('054789321', 'Driver', 41000, '2019-10-05', 'Female', '1986-01-30', '
    sophie.turner@nexstore.io', 'Sophie', 'Turner', '+96100000043', '
    Lebanon', 'South', 'Sidon', 'Bay St', 2, 6, '+96133333333', NULL, '
    Logistics'),
('023459876', 'Driver', 43000, '2015-05-22', 'Male', '1978-04-12', 'david
    .lee@nexstore.io', 'David', 'Lee', '+96100000044', 'Lebanon', 'Beirut'
    , 'Beirut', 'West Blvd', 5, 2, '+96111111111', NULL, 'Logistics');

-- Step 4: Update departments to set Employee_SSN (manager's SSN)
UPDATE Department
SET
  Employee_SSN = CASE Name
    WHEN 'Sales' THEN '123456789'
    WHEN 'Marketing' THEN '987654321'
    WHEN 'HR' THEN '123459876'
    WHEN 'Finance' THEN '567894321'
    WHEN 'Operations' THEN '543216789'
    WHEN 'IT' THEN '678912345'
    WHEN 'Customer Support' THEN '876543219'
```

```
    WHEN 'Logistics' THEN '345678912'
    WHEN 'Legal' THEN '456789123'
    WHEN 'Training' THEN '789123456'
    WHEN 'Facilities' THEN '987123456'
    WHEN 'R&D' THEN '234567891'
    ELSE Employee_SSN
  END;

-- Step 5: Update branches to set Employee_SSN
UPDATE Branch
SET
  Employee_SSN = CASE Phone_number
    WHEN '+96111111111' THEN '123456789'
    WHEN '+96122222222' THEN '987654321'
    WHEN '+96133333333' THEN '543216789'
    WHEN '+96144444444' THEN '123459876'
    WHEN '+96155555555' THEN '567894321'
    WHEN '+96166666666' THEN '678912345'
    WHEN '+96177777777' THEN '876543219'
    WHEN '+96188888888' THEN '345678912'
    WHEN '+96199999999' THEN '456789123'
    WHEN '+96112345678' THEN '234567891'
    WHEN '+96124681357' THEN '789123456'
    WHEN '+96165432198' THEN '654321987'
    WHEN '+96111223344' THEN '987123456'
    ELSE Employee_SSN
  END;

COMMIT;
```

Code Snippet 11.33 : *Insert into Branch, Department, and Employee Tables*

11.1.3.5 Driver

The following script inserts data into the Driver table.

```
INSERT INTO
  Driver (License_number, Driving_experience_years, License_expiry_date,
     Employee_SSN)
VALUES
  ('DL1001', 5, '2025-12-31', '023456789'),
  ('DL1002', 3, '2026-06-30', '087654321'),
  ('DL1003', 10, '2027-04-15', '067894321'),
  ('DL1004', 7, '2028-01-10', '043216789'),
  ('DL1005', 2, '2026-08-20', '078912345'),
  ('DL1006', 6, '2025-11-25', '076543219'),
  ('DL1007', 8, '2029-09-09', '045678912'),
  ('DL1008', 12, '2025-05-05', '056789123'),
  ('DL1009', 4, '2026-07-18', '034567891'),
```

```
  ('DL1010', 15, '2030-03-30', '089123456'),
  ('DL1011', 1, '2024-12-15', '054321987'),
  ('DL1012', 9, '2027-02-14', '021987654'),
  ('DL1013', 14, '2031-06-11', '087123456'),
  ('DL1014', 11, '2029-12-21', '054789321'),
  ('DL1015', 13, '2026-10-07', '023459876');
```

Code Snippet 11.34 : *Insert into Driver Table*

11.1.3.6 Customer

The following script inserts data into the Customer table.

```
INSERT INTO
  Customer (Phone_number, Email, First_name, Last_name, Gender,
     Registration_date, Password_hashed, Date_of_birth, Country, State,
     City, Street, Building, Apartment)
VALUES
  ('+96134567890', 'john.doe@gmail.com', 'John', 'Doe', 'Male', '2023-02-14
     ', '********', '1990-03-05', 'Lebanon', 'Beirut', 'Beirut', 'Hamra',
     12, 2),
  ('+96198765432', 'jane.smith@yahoo.com', 'Jane', 'Smith', 'Female', '
     2022-06-12', '********', '1985-01-05', 'Lebanon', 'North', 'Tripoli',
     'Mina Street', 4, 5),
  ('+96156789014', 'alice.brown@outlook.com', 'Alice', 'Brown', 'Female', '
     2023-03-01', '********', '1992-01-10', 'Lebanon', 'South', 'Sidon', '
     Corniche', 3, 6),
  ('+96178901237', 'bob.jones@hotmail.com', 'Bob', 'Jones', 'Male', '
     2021-07-18', '********', '1995-04-15', 'Lebanon', 'Mount Lebanon', '
     Jounieh', 'Coastal Rd', 6, 7),
  ('+96167890123', 'charlie.evans@aol.com', 'Charlie', 'Evans', 'Male', '
     2022-05-30', '********', '1993-02-20', 'Lebanon', 'Beirut', 'Achrafieh
     ', 'Armenia St', 9, 11),
  ('+96189012348', 'diana.lee@icloud.com', 'Diana', 'Lee', 'Female', '
     2019-11-10', '********', '1989-07-12', 'Lebanon', 'Beqaa', 'Zahle', '
     Beka St', 4, 3),
  ('+96123456789', 'frank.wilson@protonmail.com', 'Frank', 'Wilson', 'Male'
     , '2024-01-01', '********', '1993-01-17', 'Lebanon', 'Mount Lebanon',
     'Byblos', 'Roman Street', 5, 9),
  ('+96121234569', 'emma.white@gmail.com', 'Emma', 'White', 'Female', '
     2023-11-15', '********', '1985-12-25', 'Lebanon', 'South', 'Tyre', '
     Port Road', 2, 1),
  ('+96156789012', 'george.king@live.com', 'George', 'King', 'Male', '
     2022-12-25', '********', '2007-07-03', 'Lebanon', 'Mount Lebanon', '
     Antelias', 'Highway Rd', 8, 9),
  ('+96178901234', 'jack.miller@yahoo.com', 'Jack', 'Miller', 'Male', '
     2024-02-09', '********', '1992-04-17', 'Lebanon', 'Beqaa', 'Baalbek',
     'Temple Rd', 4, 3),
```

```
  ('+96189012345', 'isabella.taylor@outlook.com', 'Isabella', 'Taylor', '
     Female', '2023-07-02', '********', '1998-11-20', 'Lebanon', 'Beirut',
     'Downtown', 'Downtown', 3, 11),
  ('+96121234567', 'kevin.harris@icloud.com', 'Kevin', 'Harris', 'Male', '
     2021-04-22', '********', '1996-11-15', 'Lebanon', 'Beirut', 'Hamra', '
     Hamra', 4, 8),
  ('+96198765431', 'mike.anderson@protonmail.com', 'Mike', 'Anderson', '
     Male', '2023-05-03', '********', '1998-01-11', 'Lebanon', 'Mount
     Lebanon', 'Choueifat', 'Railway Rd', 7, 3);
```

Code Snippet 11.35 : *Insert into Customer Table*

11.1.3.7 Order

The following script inserts data into the Order table.

```
INSERT INTO
  Orders (Order_id, Notes, Payment_method, Total_amount, Is_online,
     Employee_SSN, Customer_phone_number, Date, Driver_license_number)
VALUES
  ('O101', 'Expedited delivery', 'Credit Card', 150, true, '123456789', '
     +96134567890', '2024-10-01', 'DL1001'),
  ('O102', 'Gift wrap included', 'Cash', 200, false, '111111111', '
     +96198765432', '2024-09-15', 'DL1002'),
  ('O103', 'Deliver before 5 PM', 'PayPal', 75, true, '111111111', '
     +96156789014', '2024-07-12', 'DL1003'),
  ('O104', 'Call on arrival', 'Credit Card', 300, false, '111111111', '
     +96178901237', '2024-04-07', 'DL1004'),
  ('O105', 'Special instructions', 'Apple Pay', 500, true, '111111111', '
     +96167890123', '2024-11-05', 'DL1005'),
  ('O106', 'Holiday gift', 'Credit Card', 1000, true, '111111112', '
     +96189012348', '2024-11-22', 'DL1006'),
  ('O107', 'Contactless delivery', 'PayPal', 250, true, '111111112', '
     +96123456789', '2024-07-12', 'DL1007'),
  ('O108', 'Scheduled for 3 PM', 'Credit Card', 150, false, '111111112', '
     +96121234569', '2024-06-20', 'DL1008'),
  ('O109', 'Express delivery', 'Cash', 500, false, '111111113', '
     +96156789012', '2024-11-09', 'DL1009'),
  ('O110', 'New Years package', 'Apple Pay', 750, true, '111111113', '
     +96178901234', '2024-2-03', 'DL1010'),
  ('O111', 'First-time discount', 'PayPal', 65, true, '111111113', '
     +96189012345', '2024-10-11', 'DL1011'),
  ('O112', 'VIP priority', 'Credit Card', 800, true, '654321987', '
     +96121234567', '2024-07-14', 'DL1012'),
  ('O113', 'Happy Birthday!', 'Apple Pay', 180, false, '654321987', '
     +96198765431', '2024-03-03', 'DL1013'),
  ('O114', 'Clearance sale', 'PayPal', 450, false, '654321987', '
     +96134567890', '2024-11-05', 'DL1014'),
```

```
  ('O115', 'Loyalty customer', 'Cash', 300, false, '111111114', '
     +96189012345', '2024-04-22', 'DL1015');
```

Code Snippet 11.36 : *Insert into Order Table*

11.1.3.8 Purchased

The following script inserts data into the Purchased table.

```
INSERT INTO
  Purchased (Product_SKU, Order_id, Quantity, Amount)
VALUES
  (1001, 'O101', 2, 1200),
  (1002, 'O102', 1, 1200),
  (1003, 'O103', 1, 150),
  (1004, 'O104', 2, 200),
  (1005, 'O105', 1, 300),
  (1006, 'O106', 1, 800),
  (1007, 'O107', 1, 400),
  (1008, 'O108', 3, 210),
  (1009, 'O109', 5, 125),
  (1010, 'O110', 2, 1000),
  (1011, 'O111', 1, 50),
  (1012, 'O112', 2, 60),
  (1013, 'O113', 1, 150),
  (1014, 'O114', 3, 120),
  (1015, 'O115', 1, 500);
```

Code Snippet 11.37 : *Insert into Purchased Table*

11.1.3.9 Reviews

The following script inserts data into the Reviews table.

```
INSERT INTO
  Reviews (Product_SKU, Customer_phone_number, Review_date, Rating, Comment
     , Description)
VALUES
  (1001, '+96134567890', '2024-09-01', 5, 'Great phone!', 'Excellent
     performance'),
  (1002, '+96198765432', '2024-08-20', 4, 'Good value', 'Worth the price'),
  (1003, '+96156789014', '2024-07-10', 3, 'Comfortable chair', 'Could be
     sturdier'),
  (1004, '+96178901237', '2024-06-05', 5, 'Love these shoes', 'Perfect fit'
     ),
  (1005, '+96167890123', '2024-05-15', 4, 'Nice sound', 'Great for
     beginners'),
  (1006, '+96189012348', '2024-04-22', 5, 'Amazing fridge', 'Lots of space'
     ),
```

```
  (1007, '+96198765431', '2024-03-30', 4, 'Clear picture', 'Excellent for
     gaming'),
  (1008, '+96123456789', '2024-02-18', 3, 'Good blender', 'Noisy motor'),
  (1009, '+96121234569', '2024-01-11', 5, 'Comfortable T-shirt', 'Soft
     fabric'),
  (1010, '+96156789012', '2024-09-25', 4, 'Nice smartwatch', 'Battery life
     could be better'),
  (1011, '+96178901234', '2024-08-05', 5, 'Informative book', 'Highly
     recommended'),
  (1012, '+96189012345', '2024-07-22', 4, 'Useful lamp', 'Bright and
     adjustable'),
  (1013, '+96198765431', '2024-06-12', 5, 'Excellent headphones', 'Superb
     sound quality'),
  (1014, '+96121234567', '2024-05-02', 4, 'Great soccer ball', 'Durable
     material'),
  (1015, '+96198765431', '2024-03-28', 5, 'Fantastic console', 'Best for
     gaming enthusiasts');
```

Code Snippet 11.38 : *Insert into Reviews Table*

11.1.3.10 Support Ticket

The following script inserts data into the Support Ticket table.

```
INSERT INTO
  Support_Ticket (Ticket_id, Description, Subject, Status, Priority,
     Employee_SSN, Customer_phone_number)
VALUES
  ('T001', 'Issue with product delivery', 'Delivery Issue', 'Open', 'High',
      '678912345', '+96134567890'),
  ('T002', 'Request for refund', 'Refund Request', 'Closed', 'Medium', '
     678912345', '+96198765432'),
  ('T003', 'Product not working', 'Defective Product', 'Open', 'High', '
     678912345', '+96156789014'),
  ('T004', 'Inquiry about order status', 'Order Inquiry', 'Resolved', 'Low'
     , '678912345', '+96178901237'),
  ('T005', 'Delayed shipment', 'Shipment Delay', 'Open', 'Medium', '
     678912345', '+96167890123'),
  ('T006', 'Cancel order request', 'Order Cancellation', 'Closed', 'Medium'
     , '678912345', '+96189012348'),
  ('T007', 'Issue with payment', 'Payment Issue', 'Resolved', 'High', '
     678912345', '+96123456789'),
  ('T008', 'Exchange request', 'Product Exchange', 'Open', 'Medium', '
     678912345', '+96121234569'),
  ('T009', 'Warranty inquiry', 'Warranty Inquiry', 'Resolved', 'Low', '
     678912345', '+96123456789'),
  ('T010', 'Complaint about service', 'Service Complaint', 'Open', 'High',
     '678912345', '+96156789012'),
```

```
  ('T011', 'Missing items in order', 'Missing Items', 'Open', 'Medium', '
     678912345', '+96189012345'),
  ('T012', 'Subscription issue', 'Subscription Problem', 'Resolved', 'Low',
      '678912345', '+96178901237'),
  ('T013', 'Wrong product delivered', 'Wrong Product', 'Open', 'High', '
     678912345', '+96167890123'),
  ('T014', 'Feedback submission', 'Customer Feedback', 'Closed', 'Low', '
     678912345', '+96198765431'),
  ('T015', 'Request for discount', 'Discount Request', 'Open', 'Medium', '
     678912345', '+96178901234');
```

Code Snippet 11.39 : *Insert into Support Ticket Table*

11.1.3.11 Wishlist

The following script inserts data into the Wishlist table.

```
INSERT INTO
  Wishlist (Product_SKU, Customer_phone_number, Total_amount)
VALUES
  (1001, '+96134567890', 600),
  (1002, '+96198765432', 1200),
  (1003, '+96156789014', 150),
  (1004, '+96178901237', 100),
  (1005, '+96167890123', 300),
  (1006, '+96189012348', 800),
  (1007, '+96123456789', 400),
  (1008, '+96121234569', 210),
  (1009, '+96156789012', 125),
  (1010, '+96178901234', 1000),
  (1011, '+96189012345', 50),
  (1012, '+96121234567', 60),
  (1013, '+96198765431', 150);
```

Code Snippet 11.40 : *Insert into Wishlist Table*

11.1.3.12 Working hours

The following script inserts data into the Working hours table.

```
INSERT INTO
  Working_hours (Branch_phone_number, Day, Opening_hour, Closing_hour)
VALUES
  ('+96111111111', 'Monday', '09:00', '17:00'),
  ('+96111111111', 'Tuesday', '09:00', '17:00'),
  ('+96111111111', 'Wednesday', '09:00', '17:00'),
  ('+96111111111', 'Thursday', '09:00', '17:00'),
  ('+96111111111', 'Friday', '09:00', '14:00'),
  ('+96111111111', 'Saturday', '09:00', '13:00'),
```

```
    ('+96111111111', 'Sunday', NULL, NULL),
    ('+96122222222', 'Monday', '08:00', '18:00'),
    ('+96122222222', 'Tuesday', '08:00', '18:00'),
    ('+96122222222', 'Wednesday', '08:00', '18:00'),
    ('+96122222222', 'Thursday', '08:00', '18:00'),
    ('+96122222222', 'Friday', '08:00', '13:00'),
    ('+96122222222', 'Saturday', NULL, NULL),
    ('+96122222222', 'Sunday', NULL, NULL),
    ('+96133333333', 'Monday', '10:00', '19:00'),
    ('+96133333333', 'Tuesday', '10:00', '19:00'),
    ('+96133333333', 'Wednesday', '10:00', '19:00'),
    ('+96133333333', 'Thursday', '10:00', '19:00'),
    ('+96133333333', 'Friday', '10:00', '15:00'),
    ('+96133333333', 'Saturday', '10:00', '14:00'),
    ('+96133333333', 'Sunday', NULL, NULL),
    ('+96144444444', 'Monday', '08:30', '17:30'),
    ('+96144444444', 'Tuesday', '08:30', '17:30'),
    ('+96144444444', 'Wednesday', '08:30', '17:30'),
    ('+96144444444', 'Thursday', '08:30', '17:30'),
    ('+96144444444', 'Friday', '08:30', '13:30'),
    ('+96144444444', 'Saturday', NULL, NULL),
    ('+96144444444', 'Sunday', NULL, NULL),
    ('+96155555555', 'Monday', '09:00', '18:00'),
    ('+96155555555', 'Tuesday', '09:00', '18:00'),
    ('+96155555555', 'Wednesday', '09:00', '18:00'),
    ('+96155555555', 'Thursday', '09:00', '18:00'),
    ('+96155555555', 'Friday', '09:00', '14:00'),
    ('+96155555555', 'Saturday', NULL, NULL),
    ('+96155555555', 'Sunday', NULL, NULL),
    ('+96166666666', 'Monday', '08:00', '20:00'),
    ('+96166666666', 'Tuesday', '08:00', '20:00'),
    ('+96166666666', 'Wednesday', '08:00', '20:00'),
    ('+96166666666', 'Thursday', '08:00', '20:00'),
    ('+96166666666', 'Friday', '08:00', '13:00'),
    ('+96166666666', 'Saturday', '09:00', '14:00'),
    ('+96166666666', 'Sunday', NULL, NULL),
    ('+96177777777', 'Monday', '09:30', '18:30'),
    ('+96177777777', 'Tuesday', '09:30', '18:30'),
    ('+96177777777', 'Wednesday', '09:30', '18:30'),
    ('+96177777777', 'Thursday', '09:30', '18:30'),
    ('+96177777777', 'Friday', '09:30', '14:30'),
    ('+96177777777', 'Saturday', NULL, NULL),
    ('+96177777777', 'Sunday', NULL, NULL),
    ('+96188888888', 'Monday', '09:00', '17:00'),
    ('+96188888888', 'Tuesday', '09:00', '17:00'),
    ('+96188888888', 'Wednesday', '09:00', '17:00'),
    ('+96188888888', 'Thursday', '09:00', '17:00'),
    ('+96188888888', 'Friday', '09:00', '13:00'),
```

```
  ('+96188888888', 'Saturday', NULL, NULL),
  ('+96188888888', 'Sunday', NULL, NULL),
  ('+96199999999', 'Monday', '08:00', '19:00'),
  ('+96199999999', 'Tuesday', '08:00', '19:00'),
  ('+96199999999', 'Wednesday', '08:00', '19:00'),
  ('+96199999999', 'Thursday', '08:00', '19:00'),
  ('+96199999999', 'Friday', '08:00', '14:00'),
  ('+96199999999', 'Saturday', '09:00', '13:00'),
  ('+96199999999', 'Sunday', NULL, NULL),
  ('+96112345678', 'Monday', '09:00', '18:00'),
  ('+96112345678', 'Tuesday', '09:00', '18:00'),
  ('+96112345678', 'Wednesday', '09:00', '18:00'),
  ('+96112345678', 'Thursday', '09:00', '18:00'),
  ('+96112345678', 'Friday', '09:00', '13:00'),
  ('+96112345678', 'Saturday', NULL, NULL),
  ('+96112345678', 'Sunday', NULL, NULL),
  ('+96124681357', 'Monday', '08:30', '18:30'),
  ('+96124681357', 'Tuesday', '08:30', '18:30'),
  ('+96124681357', 'Wednesday', '08:30', '18:30'),
  ('+96124681357', 'Thursday', '08:30', '18:30'),
  ('+96124681357', 'Friday', '08:30', '14:30'),
  ('+96124681357', 'Saturday', NULL, NULL),
  ('+96124681357', 'Sunday', NULL, NULL),
  ('+96165432198', 'Monday', '09:00', '19:00'),
  ('+96165432198', 'Tuesday', '09:00', '19:00'),
  ('+96165432198', 'Wednesday', '09:00', '19:00'),
  ('+96165432198', 'Thursday', '09:00', '19:00'),
  ('+96165432198', 'Friday', '09:00', '14:00'),
  ('+96165432198', 'Saturday', '10:00', '14:00'),
  ('+96165432198', 'Sunday', NULL, NULL),
  ('+96111223344', 'Monday', '08:00', '17:00'),
  ('+96111223344', 'Tuesday', '08:00', '17:00'),
  ('+96111223344', 'Wednesday', '08:00', '17:00'),
  ('+96111223344', 'Thursday', '08:00', '17:00'),
  ('+96111223344', 'Friday', '08:00', '13:00'),
  ('+96111223344', 'Saturday', NULL, NULL),
  ('+96111223344', 'Sunday', NULL, NULL);
```

Code Snippet 11.41 : *Insert into Working Hours Table*

11.1.3.13 Dependent

The following script inserts data into the Dependent table.

```
INSERT INTO
  Dependent (Employee_SSN, Name, Gender, Date_of_birth, Relationship)
VALUES
  ('123456789', 'Sarah Doe', 'Female', '2015-04-15', 'Daughter'),
  ('987654321', 'James Smith', 'Male', '2013-07-20', 'Son'),
```

```
  ('333333333', 'Emily Brown', 'Female', '2017-02-28', 'Daughter'),
  ('333333333', 'Lucas Jones', 'Male', '2018-11-05', 'Son'),
  ('444444444', 'Olivia Evans', 'Female', '2016-09-12', 'Daughter'),
  ('543216789', 'Ethan White', 'Male', '2020-03-22', 'Son'),
  ('666666666', 'Chloe Lee', 'Female', '2014-08-01', 'Daughter'),
  ('876543219', 'Liam Harris', 'Male', '2015-12-15', 'Son'),
  ('345678912', 'Mia Taylor', 'Female', '2018-10-03', 'Daughter'),
  ('888888881', 'Noah Wilson', 'Male', '2021-06-08', 'Son'),
  ('888888882', 'Sophia Martin', 'Female', '2019-05-25', 'Daughter'),
  ('789123456', 'Benjamin Scott', 'Male', '2012-01-19', 'Son'),
  ('999999991', 'Emma Anderson', 'Female', '2015-07-13', 'Daughter'),
  ('999999992', 'Mason Miller', 'Male', '2018-03-09', 'Son'),
  ('999999992', 'Ava Thomas', 'Female', '2016-02-04', 'Daughter');
```

Code Snippet 11.42 : *Insert into Dependent Table*

11.1.3.14 Product Image URLs

The following script inserts data into the Product Image URLs table.

```
INSERT INTO
  Product_Image_URLs (Product_SKU, Product_Image_URL)
VALUES
  ('1001', 'https://nextstore.io/images/product/1.jpg'),
  ('1002', 'https://nextstore.io/images/product/2.jpg'),
  ('1003', 'https://nextstore.io/images/product/3.jpg'),
  ('1004', 'https://nextstore.io/images/product/4.jpg'),
  ('1005', 'https://nextstore.io/images/product/5.jpg'),
  ('1006', 'https://nextstore.io/images/product/6.jpg'),
  ('1007', 'https://nextstore.io/images/product/7.jpg'),
  ('1008', 'https://nextstore.io/images/product/8.jpg'),
  ('1009', 'https://nextstore.io/images/product/9.jpg'),
  ('1010', 'https://nextstore.io/images/product/10.jpg'),
  ('1011', 'https://nextstore.io/images/product/11.jpg'),
  ('1012', 'https://nextstore.io/images/product/12.jpg'),
  ('1013', 'https://nextstore.io/images/product/13.jpg'),
  ('1014', 'https://nextstore.io/images/product/14.jpg'),
  ('1015', 'https://nextstore.io/images/product/15.jpg');
```

Code Snippet 11.43 : *Insert into Product Image URLs Table*

11.1.3.15 Review Image URLs

The following script inserts data into the Review Image URLs table.

```
INSERT INTO
  Review_Image_URLs (Product_SKU, Customer_phone_number, Product_Image_URL)
VALUES
  ('1001', '+96134567890', 'https://nextstore.io/images/image1.jpg'),
```

```
  ('1002', '+96198765432', 'https://nextstore.io/images/image2.jpg'),
  ('1003', '+96156789014', 'https://nextstore.io/images/image3.jpg'),
  ('1004', '+96178901237', 'https://nextstore.io/images/image4.jpg'),
  ('1005', '+96167890123', 'https://nextstore.io/images/image5.jpg'),
  ('1006', '+96189012348', 'https://nextstore.io/images/image6.jpg'),
  ('1007', '+96123456789', 'https://nextstore.io/images/image7.jpg'),
  ('1008', '+96121234569', 'https://nextstore.io/images/image8.jpg'),
  ('1009', '+96156789012', 'https://nextstore.io/images/image9.jpg'),
  ('1010', '+96178901234', 'https://nextstore.io/images/image10.jpg'),
  ('1011', '+96189012345', 'https://nextstore.io/images/image11.jpg'),
  ('1012', '+96121234567', 'https://nextstore.io/images/image12.jpg'),
  ('1013', '+96198765431', 'https://nextstore.io/images/image13.jpg'),
  ('1014', '+96198765432', 'https://nextstore.io/images/image14.jpg'),
  ('1015', '+96123456789', 'https://nextstore.io/images/image15.jpg');
```

Code Snippet 11.44 : *Insert into Review Image URLs Table*

11.1.3.16 Located in

The following script inserts data into the Located in table.

```
INSERT INTO
  Located_in (Product_SKU, Branch_phone_number, Quantity, Shelf_location)
VALUES
  ('1001', '+96111111111', 50, 'A1'),
  ('1002', '+96122222222', 30, 'B2'),
  ('1003', '+96133333333', 20, 'C3'),
  ('1004', '+96144444444', 15, 'D4'),
  ('1005', '+96155555555', 60, 'E5'),
  ('1006', '+96166666666', 25, 'F6'),
  ('1007', '+96177777777', 10, 'G7'),
  ('1008', '+96188888888', 35, 'H8'),
  ('1009', '+96199999999', 40, 'I9'),
  ('1010', '+96112345678', 50, 'J10'),
  ('1011', '+96124681357', 70, 'K11'),
  ('1012', '+96124681357', 20, 'L12'),
  ('1013', '+96165432198', 15, 'M13'),
  ('1014', '+96165432198', 5, 'N14'),
  ('1015', '+96111223344', 55, 'O15');
```

Code Snippet 11.45 : *Insert into Located In Table*

11.1.3.17 Colors

The following script inserts data into the Colors table.

```
INSERT INTO
  Colors (Product_SKU, Product_color)
VALUES
```

```
  ('1001', 'Red'),
  ('1002', 'Blue'),
  ('1003', 'Green'),
  ('1004', 'Black'),
  ('1005', 'White'),
  ('1006', 'Yellow'),
  ('1007', 'Pink'),
  ('1008', 'Purple'),
  ('1009', 'Orange'),
  ('1010', 'Brown'),
  ('1011', 'Gray'),
  ('1012', 'Gold'),
  ('1013', 'Silver'),
  ('1014', 'Beige'),
  ('1015', 'Navy');
```

Code Snippet 11.46 : *Insert into Colors Table*

11.1.3.18 Coupon

The following script inserts data into the Coupon table.

```
INSERT INTO
  Coupon (Code, Description, Discount_percent, Times_used,
     Minimum_order_amount, Maximum_order_amount, Usage_limit, Valid_from,
     Valid_to, Order_ID, Discount_amount, Redeem_date)
VALUES
  ('DIS10', '10% off', 10, 25, 50, 500, 100, '2024-01-01', '2024-12-31', '
     O101', 5, '2024-10-01'),
  ('DIS20', '20% off', 20, 40, 100, 1000, 75, '2024-01-01', '2024-12-31', '
     O102', 10, '2024-09-15'),
  ('SAVE15', '15% off', 15, 50, 200, 1000, 25, '2024-01-01', '2024-12-31',
     'O104', 15, '2024-07-07'),
  ('BOGO', 'Buy 1 Get 1', 50, 20, 100, 500, 50, '2024-01-01', '2024-12-31',
      'O105', 20, '2024-06-06'),
  ('HOLIDAY50', '50% off', 50, 15, 200, 1000, 30, '2024-01-01', '2024-08-01
     ', 'O106', 50, '2024-04-01'),
  ('FLASH5', '5% off', 5, 150, 25, 250, 50, '2024-01-01', '2024-12-31', '
     O107', 2, '2024-07-20'),
  ('SUMMER30', '30% off', 30, 80, 150, 1500, 30, '2024-01-01', '2024-12-31'
     , 'O108', 45, '2024-09-20'),
  ('BLCKFRIDAY', '40% off', 40, 100, 300, 2000, 75, '2024-01-01', '
     2024-12-31', 'O109', 60, '2024-11-01'),
  ('NEWYEAR25', '25% off', 25, 70, 100, 1000, 50, '2024-01-01', '2024-12-31
     ', 'O110', 50, '2024-10-31'),
  ('WELCOME', '10% for new users', 10, 10, 50, 500, 25, '2024-01-01', '
     2024-12-31', 'O111', 5, '2024-11-15'),
  ('VIP20', '20% VIP discount', 20, 30, 150, 1500, 50, '2024-01-01', '
     2024-12-31', 'O112', 30, '2024-05-01'),
```

```
  ('BIRTHDAY', '25% off on birthday', 25, 20, 100, 1000, 25, '2024-01-01',
     '2024-12-31', 'O113', 10, '2024-04-10'),
  ('CLEARANCE', 'Up to 70% off', 70, 50, 500, 5000, 100, '2024-01-01', '
     2024-11-15', 'O114', 350, '2024-05-14'),
  ('LOYALTY', '15% off for loyal customers', 15, 15, 50, 1000, 60, '
     2024-01-01', '2024-12-31', 'O115', 15, '2024-04-22');
```

Code Snippet 11.47 : *Insert into Coupon Table*

11.1.3.19 Department location

The following script inserts data into the Department location table.

```
INSERT INTO
  Department_location (Department_name, Location)
VALUES
  ('Sales', 'Beirut'),
  ('Marketing', 'Tripoli'),
  ('HR', 'Zahle'),
  ('Finance', 'Jounieh'),
  ('Operations', 'Sidon'),
  ('IT', 'Byblos'),
  ('Customer Support', 'Tyre'),
  ('Logistics', 'Baalbek'),
  ('Legal', 'Batroun'),
  ('Training', 'Dora'),
  ('Facilities', 'Antelias'),
  ('R&D', 'Achrafieh');
```

Code Snippet 11.48 : *Insert into Department Location Table*

11.1.4 Select Queries

11.1.4.1 Supplier

The following script selects data from the Supplier table.

```
SELECT * FROM Supplier;
```

Code Snippet 11.49 : *Select from Supplier Table*

Query Query History

```
1  SELECT * FROM Supplier;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No: 1 of 1

	website [PK] character varying (50)	supplier_name character varying (50)	contact_person_email character varying (50)	contact_person_first_name character varying (50)	contact_person_last_name character varying (50)	contact_person_phone_number character varying (25)
1	www.techbrand.com	TechBrand	contact@techbrand.com	Alice	Doe	+96134567890
2	www.computex.com	ComputeX	support@computex.com	John	Smith	+96198765432
3	www.comfortco.com	ComfortCo	info@comfortco.com	Emma	Johnson	+96145678901
4	www.sportwear.com	SportWear	sales@sportwear.com	Michael	Brown	+96167890123
5	www.musicpro.com	MusicPro	music@musicpro.com	Sarah	Lee	+96178901234
6	www.homeappl.com	HomeAppl	appliances@homeappl.com	David	Evans	+96189012345
7	www.visionco.com	VisionCo	vision@visionco.com	Jessica	Taylor	+96190123456
8	www.kitchenx.com	KitchenX	service@kitchenx.com	Kevin	Wilson	+96101234567
9	www.fashionhub.com	FashionHub	hello@fashionhub.com	Emily	Harris	+96123456789
10	www.timekeep.com	TimeKeep	contact@timekeep.com	Daniel	Martin	+96156789012
11	www.edubooks.com	EduBooks	edu@edubooks.com	Olivia	White	+96167890124
12	www.lightpro.com	LightPro	light@lightpro.com	Robert	Scott	+96178901235
13	www.audiomax.com	AudioMax	audio@audiomax.com	Sophia	Anderson	+96189012346
14	www.gamezone.com	GameZone	games@gamezone.com	Liam	Thomas	+96190123457
15	www.sportwearlb.com	SportWearLebanon	store@sportwear.com	Noah	Miller	+96101234568

Figure 11.1: *Select from Supplier Table*

11.1.4.2 Category

The following script selects data from the Category table.

```
1  SELECT * FROM Category;
```

Code Snippet 11.50 : *Select from Category Table*

Query Query History

```
1  SELECT * FROM Category;
```

Data Output Messages Notifications

	name [PK] character varying (50)	description text	parent_category_name character varying (50)
1	Electronics	Devices and gadgets	[null]
2	Clothing	Apparel and fashion items	[null]
3	Furniture	Home and office furniture	[null]
4	Fashion	Clothing and accessories	[null]
5	Books	Printed and digital books	Stationery
6	Groceries	Food and daily supplies	Health
7	Sports	Sporting equipment and apparel	Clothing
8	Beauty	Cosmetics and skincare products	Health
9	Toys	Toys for kids and adults	Sports
10	Automotive	Car parts and accessories	Electronics
11	Jewelry	Watches, rings, and necklaces	Fashion
12	Health	Medical supplies and equipment	Beauty
13	Stationery	Office and school supplies	Furniture
14	Pets	Pet food and accessories	Groceries
15	Music	Instruments and music equipment	Art
16	Art	Art supplies and crafts	Stationery

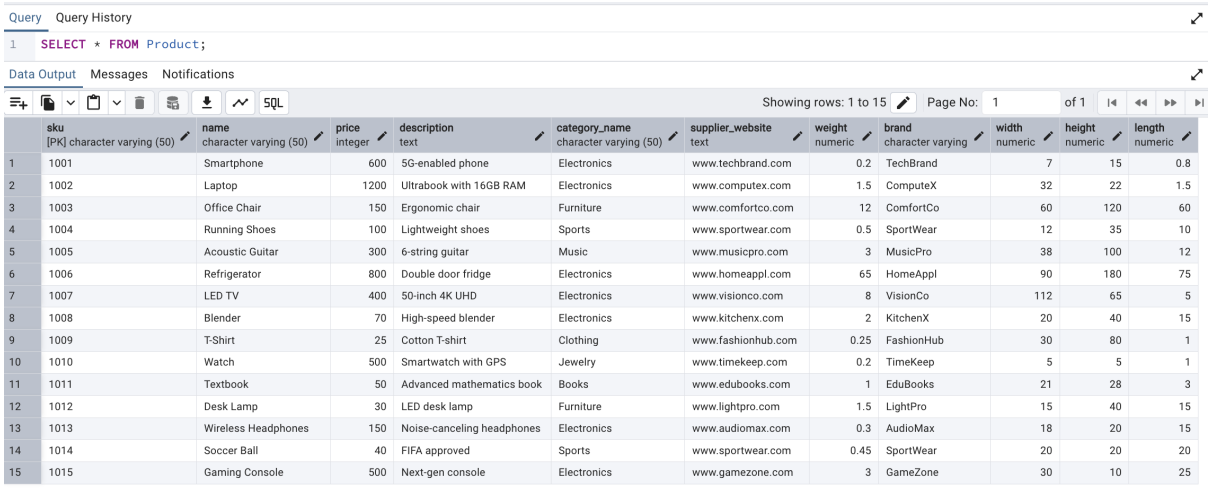
Figure 11.2: *Select from Category Table*

11.1.4.3 Product

The following script selects data from the Product table.

```
1  SELECT * FROM Product;
```

Code Snippet 11.51 : *Select from Product Table*



Query Query History

```
1  SELECT * FROM Product;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No: 1 of 1

	sku [PK] character varying (50)	name character varying (50)	price integer	description text	category_name character varying (50)	supplier_website text	weight numeric	brand character varying	width numeric	height numeric	length numeric
1	1001	Smartphone	600	5G-enabled phone	Electronics	www.techbrand.com	0.2	TechBrand	7	15	0.8
2	1002	Laptop	1200	Ultrabook with 16GB RAM	Electronics	www.computex.com	1.5	ComputeX	32	22	1.5
3	1003	Office Chair	150	Ergonomic chair	Furniture	www.comfortco.com	12	ComfortCo	60	120	60
4	1004	Running Shoes	100	Lightweight shoes	Sports	www.sportwear.com	0.5	SportWear	12	35	10
5	1005	Acoustic Guitar	300	6-string guitar	Music	www.musicpro.com	3	MusicPro	38	100	12
6	1006	Refrigerator	800	Double door fridge	Electronics	www.homeappl.com	65	HomeAppl	90	180	75
7	1007	LED TV	400	50-inch 4K UHD	Electronics	www.visionco.com	8	VisionCo	112	65	5
8	1008	Blender	70	High-speed blender	Electronics	www.kitchenx.com	2	KitchenX	20	40	15
9	1009	T-Shirt	25	Cotton T-shirt	Clothing	www.fashionhub.com	0.25	FashionHub	30	80	1
10	1010	Watch	500	Smartwatch with GPS	Jewelry	www.timekeep.com	0.2	TimeKeep	5	5	1
11	1011	Textbook	50	Advanced mathematics book	Books	www.edubooks.com	1	EduBooks	21	28	3
12	1012	Desk Lamp	30	LED desk lamp	Furniture	www.lightpro.com	1.5	LightPro	15	40	15
13	1013	Wireless Headphones	150	Noise-canceling headphones	Electronics	www.audiomax.com	0.3	AudioMax	18	20	15
14	1014	Soccer Ball	40	FIFA approved	Sports	www.sportwear.com	0.45	SportWear	20	20	20
15	1015	Gaming Console	500	Next-gen console	Electronics	www.gamezone.com	3	GameZone	30	10	25

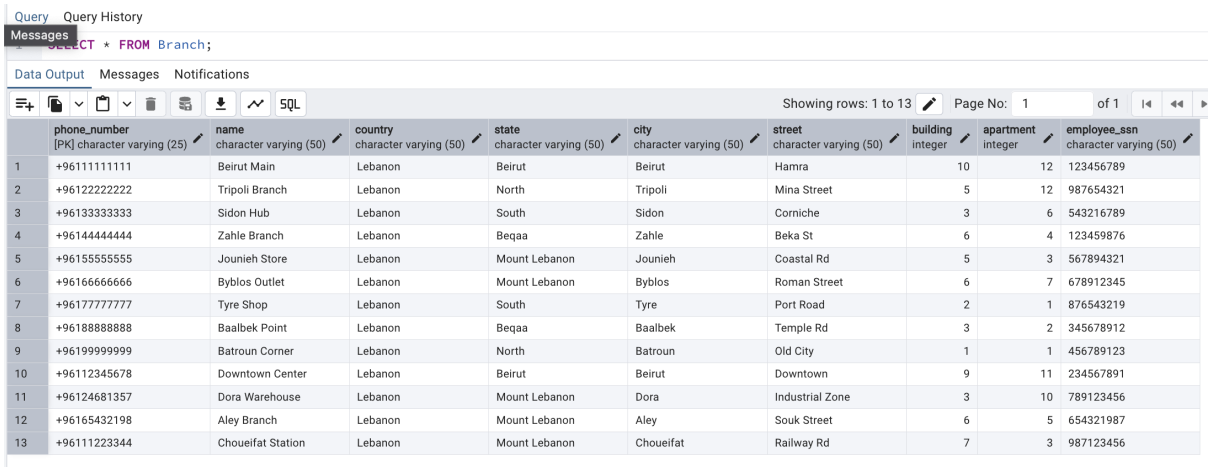
Figure 11.3: *Select from Product Table*

11.1.4.4 Branch

The following script selects data from the Branch table.

```
1  SELECT * FROM Branch;
```

Code Snippet 11.52 : *Select from Branch Table*



Query Query History

Messages

```
SELECT * FROM Branch;
```

Data Output Messages Notifications

Showing rows: 1 to 13 Page No: 1 of 1

	phone_number [PK] character varying (25)	name character varying (50)	country character varying (50)	state character varying (50)	city character varying (50)	street character varying (50)	building integer	apartment integer	employee_ssn character varying (50)
1	+96111111111	Beirut Main	Lebanon	Beirut	Beirut	Hamra	10	12	123456789
2	+96122222222	Tripoli Branch	Lebanon	North	Tripoli	Mina Street	5	12	987654321
3	+96133333333	Sidon Hub	Lebanon	South	Sidon	Corniche	3	6	543216789
4	+96144444444	Zahle Branch	Lebanon	Beqaa	Zahle	Beka St	6	4	123459876
5	+96155555555	Jounieh Store	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	3	567894321
6	+96166666666	Byblos Outlet	Lebanon	Mount Lebanon	Byblos	Roman Street	6	7	678912345
7	+96177777777	Tyre Shop	Lebanon	South	Tyre	Port Road	2	1	876543219
8	+96188888888	Baalbek Point	Lebanon	Beqaa	Baalbek	Temple Rd	3	2	345678912
9	+96199999999	Batroun Corner	Lebanon	North	Batroun	Old City	1	1	456789123
10	+96112345678	Downtown Center	Lebanon	Beirut	Beirut	Downtown	9	11	234567891
11	+96124681357	Dora Warehouse	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	10	789123456
12	+96165432198	Aley Branch	Lebanon	Mount Lebanon	Aley	Souk Street	6	5	654321987
13	+96111223344	Choueifat Station	Lebanon	Mount Lebanon	Choueifat	Railway Rd	7	3	987123456

Figure 11.4: *Select from Branch Table*

11.1.4.5 Department

The following script selects data from the Department table.

```
1  SELECT * FROM Department;
```

Code Snippet 11.53 : *Select from Department Table*

Query Query History

```
1  SELECT * FROM Department;
```

Data Output Messages Notifications

Showing

	name [PK] character varying (50)	number_of_employees integer	employee_ssn character varying (50)	manager_start_date date
1	Sales	6	123456789	2022-03-01
2	Marketing	2	987654321	2021-06-15
3	HR	2	123459876	2023-01-10
4	Finance	3	567894321	2019-10-05
5	Operations	2	543216789	2020-08-25
6	IT	1	678912345	2024-04-18
7	Customer Support	2	876543219	2022-09-12
8	Logistics	3	345678912	2020-11-20
9	Legal	1	456789123	2021-02-14
10	Training	4	789123456	2023-03-27
11	Facilities	1	987123456	2019-06-01
12	R&D	1	234567891	2022-05-10

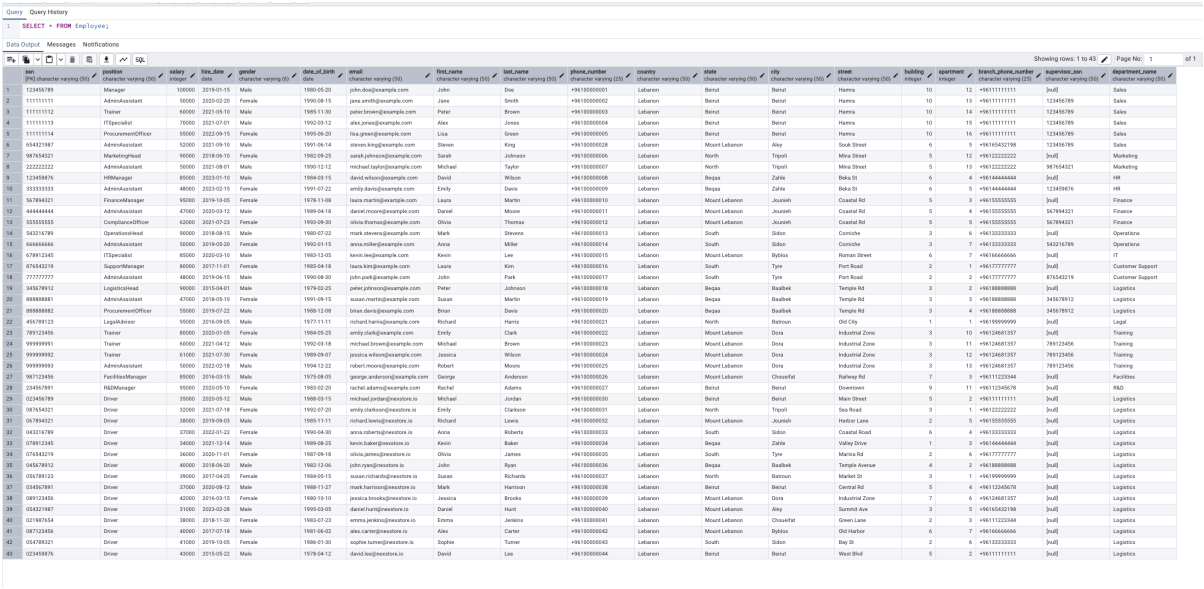
Figure 11.5: *Select from Department Table*

11.1.4.6 Employee

The following script selects data from the Employee table.

```
1  SELECT * FROM Employee;
```

Code Snippet 11.54 : *Select from Employee Table*



Query Query History

```
SELECT * FROM Employee;
```

Data Output Messages Notifications

Showing rows: 1 to 43 Page No: 1 of 1

ssn [PK] character varying (50)	position character varying (50)	salary integer	hire_date date	gender character varying (6)	date_of_birth date	email character varying (50)	first_name character varying (50)	last_name character varying (50)	phone_number character varying (25)	country character varying (50)	state character varying (50)	city character varying (50)	street character varying (50)	building integer	apartment integer	branch_phone_number character varying (25)	supervisor_ssn character varying (50)	department_name character varying (50)
123456789	Manager	100000	2019-01-15	Male	1980-05-20	john.doe@example.com	John	Doe	+96100000001	Lebanon	Beirut	Beirut	Hamra	10	12	+9611111111	[null]	Sales
111111111	AdminAssistant	50000	2020-02-20	Female	1990-08-15	jane.smith@example.com	Jane	Smith	+96100000002	Lebanon	Beirut	Beirut	Hamra	10	13	+9611111111	123456789	Sales
111111112	Trainer	60000	2021-05-10	Male	1985-11-30	peter.brown@example.com	Peter	Brown	+96100000003	Lebanon	Beirut	Beirut	Hamra	10	14	+9611111111	123456789	Sales
111111113	ITSpecialist	70000	2021-07-01	Male	1992-03-12	alex.jones@example.com	Alex	Jones	+96100000004	Lebanon	Beirut	Beirut	Hamra	10	15	+9611111111	123456789	Sales
111111114	ProcurementOfficer	55000	2022-09-15	Female	1995-06-20	lisa.green@example.com	Lisa	Green	+96100000005	Lebanon	Beirut	Beirut	Hamra	10	16	+9611111111	123456789	Sales
654321987	AdminAssistant	52000	2021-09-10	Male	1991-08-14	steven.king@example.com	Steven	King	+96100000028	Lebanon	Mount Lebanon	Aley	Souk Street	6	5	+96165432198	123456789	Sales
987654321	MarketingHead	90000	2018-06-10	Female	1982-09-25	sarah.johnson@example.com	Sarah	Johnson	+96100000006	Lebanon	North	Tripoli	Mina Street	5	12	+96122222222	[null]	Marketing
222222222	AdminAssistant	50000	2021-08-01	Male	1990-12-12	michael.taylor@example.com	Michael	Taylor	+96100000007	Lebanon	North	Tripoli	Mina Street	5	13	+96122222222	987654321	Marketing
123459876	HRManager	85000	2023-01-10	Male	1984-03-15	david.wilson@example.com	David	Wilson	+96100000008	Lebanon	Beqaa	Zahle	Beka St	6	4	+96144444444	[null]	HR
333333333	AdminAssistant	48000	2023-02-15	Female	1991-07-22	emily.davis@example.com	Emily	Davis	+96100000009	Lebanon	Beqaa	Zahle	Beka St	6	5	+96144444444	123459876	HR
567894321	FinanceManager	95000	2019-10-05	Female	1978-11-08	laura.martin@example.com	Laura	Martin	+96100000010	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	3	+96155555555	[null]	Finance
444444444	AdminAssistant	47000	2020-03-12	Male	1989-04-18	daniel.moore@example.com	Daniel	Moore	+96100000011	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	4	+96155555555	567894321	Finance
555555555	ComplianceOfficer	62000	2021-07-23	Female	1993-09-30	olivia.thomas@example.com	Olivia	Thomas	+96100000012	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	5	+96155555555	567894321	Finance
543216789	OperationsHead	90000	2018-08-15	Male	1980-07-22	mark.stevens@example.com	Mark	Stevens	+96100000013	Lebanon	South	Sidon	Comiche	3	6	+96133333333	[null]	Operations
666666666	AdminAssistant	50000	2019-05-20	Female	1992-01-15	anna.miller@example.com	Anna	Miller	+96100000014	Lebanon	South	Sidon	Comiche	3	7	+96133333333	543216789	Operations
678912345	ITSpecialist	85000	2020-03-10	Male	1983-12-05	kevin.lee@example.com	Kevin	Lee	+96100000015	Lebanon	Mount Lebanon	Byblos	Roman Street	6	7	+96166666666	[null]	IT
876543219	SupportManager	80000	2017-11-01	Female	1985-04-18	laura.kim@example.com	Laura	Kim	+96100000016	Lebanon	South	Tyre	Port Road	2	1	+96177777777	[null]	Customer Support
777777777	AdminAssistant	48000	2019-06-15	Male	1990-08-30	john.park@example.com	John	Park	+96100000017	Lebanon	South	Tyre	Port Road	2	2	+96177777777	876543219	Customer Support
345678912	LogisticsHead	90000	2015-04-01	Male	1979-02-25	peter.johnson@example.com	Peter	Johnson	+96100000018	Lebanon	Beqaa	Baalbek	Temple Rd	3	2	+96188888888	[null]	Logistics
888888881	AdminAssistant	47000	2018-05-10	Female	1991-08-15	susan.martin@example.com	Susan	Martin	+96100000019	Lebanon	Beqaa	Baalbek	Temple Rd	3	3	+96188888888	345678912	Logistics
888888882	ProcurementOfficer	55000	2019-07-22	Male	1988-12-08	brian.davis@example.com	Brian	Davis	+96100000020	Lebanon	Beqaa	Baalbek	Temple Rd	3	4	+96188888888	345678912	Logistics
456789123	LegalAdvisor	95000	2016-09-05	Male	1977-11-11	richard.harris@example.com	Richard	Harris	+96100000021	Lebanon	North	Batroun	Old City	1	1	+96199999999	[null]	Legal
789123456	Trainer	80000	2020-01-05	Female	1984-05-25	emily.clark@example.com	Emily	Clark	+96100000022	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	10	+96124681357	[null]	Training
999999991	Trainer	60000	2021-04-12	Male	1992-03-18	michael.brown@example.com	Michael	Brown	+96100000023	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	11	+96124681357	789123456	Training
999999992	Trainer	61000	2021-07-30	Female	1989-09-07	jessica.wilson@example.com	Jessica	Wilson	+96100000024	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	12	+96124681357	789123456	Training
999999993	AdminAssistant	50000	2022-02-18	Male	1994-12-22	robert.moore@example.com	Robert	Moore	+96100000025	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	13	+96124681357	789123456	Training
987123456	FacilitiesManager	85000	2016-03-15	Male	1975-08-05	george.anderson@example.com	George	Anderson	+96100000026	Lebanon	Mount Lebanon	Chouefat	Railway Rd	7	3	+96111223344	[null]	Facilities
234567891	R&DManager	95000	2020-05-10	Female	1983-02-20	rachel.adams@example.com	Rachel	Adams	+96100000027	Lebanon	Beirut	Beirut	Downtown	9	11	+96112345678	[null]	R&D
023456789	Driver	35000	2020-05-12	Male	1988-03-15	michael.jordan@nexstore.io	Michael	Jordan	+96100000030	Lebanon	Beirut	Beirut	Main Street	5	2	+96111111111	[null]	Logistics
087654321	Driver	32000	2021-07-18	Female	1992-07-20	emily.clarkson@nexstore.io	Emily	Clarkson	+96100000031	Lebanon	North	Tripoli	Sea Road	3	1	+96122222222	[null]	Logistics
067894321	Driver	38000	2019-09-03	Male	1985-11-11	richard.lewis@nexstore.io	Richard	Lewis	+96100000032	Lebanon	Mount Lebanon	Jounieh	Harbor Lane	2	5	+96155555555	[null]	Logistics
043216789	Driver	37000	2022-01-22	Female	1990-04-30	anna.roberts@nexstore.io	Anna	Roberts	+96100000033	Lebanon	South	Sidon	Coastal Road	6	4	+96133333333	[null]	Logistics
078912345	Driver	34000	2021-12-14	Male	1989-08-25	kevin.baker@nexstore.io	Kevin	Baker	+96100000034	Lebanon	Beqaa	Zahle	Valley Drive	1	3	+96144444444	[null]	Logistics
076543219	Driver	36000	2020-11-01	Female	1987-09-18	olivia.james@nexstore.io	Olivia	James	+96100000035	Lebanon	South	Tyre	Marina Rd	2	6	+96177777777	[null]	Logistics
045678912	Driver	40000	2018-06-20	Male	1982-12-06	john.ryan@nexstore.io	John	Ryan	+96100000036	Lebanon	Beqaa	Baalbek	Temple Avenue	4	2	+96188888888	[null]	Logistics
056789123	Driver	39000	2017-04-25	Female	1984-05-15	susan.richards@nexstore.io	Susan	Richards	+96100000037	Lebanon	North	Batroun	Market St	3	1	+96199999999	[null]	Logistics
034567891	Driver	37000	2020-08-12	Male	1988-11-27	mark.harrison@nexstore.io	Mark	Harrison	+96100000038	Lebanon	Beirut	Beirut	Central Rd	5	4	+96112345678	[null]	Logistics
089123456	Driver	42000	2016-03-15	Female	1980-10-10	jessica.brooks@nexstore.io	Jessica	Brooks	+96100000039	Lebanon	Mount Lebanon	Dora	Industrial Zone	7	6	+96124681357	[null]	Logistics
054321987	Driver	31000	2023-02-28	Male	1995-03-05	daniel.hunt@nexstore.io	Daniel	Hunt	+96100000040	Lebanon	Mount Lebanon	Aley	Summit Ave	3	5	+96165432198	[null]	Logistics
021987654	Driver	38000	2018-11-20	Female	1983-07-23	emma.jenkins@nexstore.io	Emma	Jenkins	+96100000041	Lebanon	Mount Lebanon	Chouefat	Green Lane	2	3	+96111223344	[null]	Logistics
087123456	Driver	40000	2017-07-18	Male	1981-06-02	alex.carter@nexstore.io	Alex	Carter	+96100000042	Lebanon	Mount Lebanon	Byblos	Old Harbor	6	7	+96166666666	[null]	Logistics
054789321	Driver	41000	2019-10-05	Female	1986-01-30	sophie.turner@nexstore.io	Sophie	Turner	+96100000043	Lebanon	South	Sidon	Bay St	2	6	+96133333333	[null]	Logistics
023459876	Driver	43000	2015-05-22	Male	1978-04-12	david.lee@nexstore.io	David	Lee	+96100000044	Lebanon	Beirut	Beirut	West Blvd	5	2	+96111111111	[null]	Logistics

Figure 11.6: *Select from Employee Table*

11.1.4.7 Driver

The following script selects data from the Driver table.

```
1 SELECT * FROM Driver;
```

Code Snippet 11.55 : *Select from Driver Table*

Query Query History

```
1  SELECT * FROM Driver;
```

Data Output Messages Notifications

Showing rows:

	license_number [PK] character varying (50)	driving_experience_years integer	license_expiry_date date	employee_ssn character varying (50)
1	DL1001	5	2025-12-31	023456789
2	DL1002	3	2026-06-30	087654321
3	DL1003	10	2027-04-15	067894321
4	DL1004	7	2028-01-10	043216789
5	DL1005	2	2026-08-20	078912345
6	DL1006	6	2025-11-25	076543219
7	DL1007	8	2029-09-09	045678912
8	DL1008	12	2025-05-05	056789123
9	DL1009	4	2026-07-18	034567891
10	DL1010	15	2030-03-30	089123456
11	DL1011	1	2024-12-15	054321987
12	DL1012	9	2027-02-14	021987654
13	DL1013	14	2031-06-11	087123456
14	DL1014	11	2029-12-21	054789321
15	DL1015	13	2026-10-07	023459876

Figure 11.7: *Select from Driver Table*

11.1.4.8 Customer

The following script selects data from the Customer table.

```
1  SELECT * FROM Customer;
```

Code Snippet 11.56 : *Select from Customer Table*

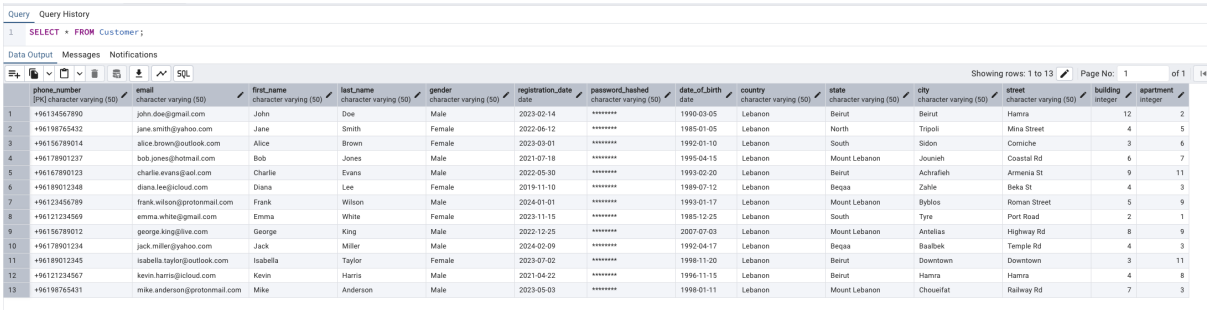


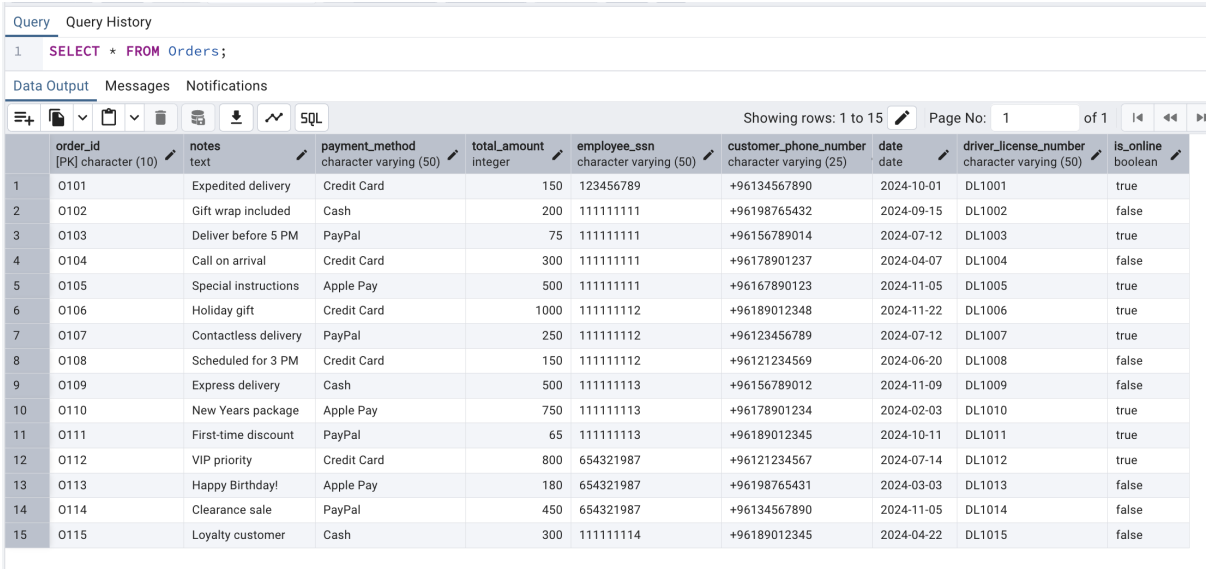
Figure 11.8: *Select from Customer Table*

11.1.4.9 Orders

The following script selects data from the Orders table.

```sql
SELECT * FROM Orders;
```

Code Snippet 11.57 : *Select from Orders Table*



Query Query History

```
1  SELECT * FROM Orders;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No: 1 of 1

	order_id [PK] character (10)	notes text	payment_method character varying (50)	total_amount integer	employee_ssn character varying (50)	customer_phone_number character varying (25)	date date	driver_license_number character varying (50)	is_online boolean
1	O101	Expedited delivery	Credit Card	150	123456789	+96134567890	2024-10-01	DL1001	true
2	O102	Gift wrap included	Cash	200	111111111	+96198765432	2024-09-15	DL1002	false
3	O103	Deliver before 5 PM	PayPal	75	111111111	+96156789014	2024-07-12	DL1003	true
4	O104	Call on arrival	Credit Card	300	111111111	+96178901237	2024-04-07	DL1004	false
5	O105	Special instructions	Apple Pay	500	111111111	+96167890123	2024-11-05	DL1005	true
6	O106	Holiday gift	Credit Card	1000	111111112	+96189012348	2024-11-22	DL1006	true
7	O107	Contactless delivery	PayPal	250	111111112	+96123456789	2024-07-12	DL1007	true
8	O108	Scheduled for 3 PM	Credit Card	150	111111112	+96121234569	2024-06-20	DL1008	false
9	O109	Express delivery	Cash	500	111111113	+96156789012	2024-11-09	DL1009	false
10	O110	New Years package	Apple Pay	750	111111113	+96178901234	2024-02-03	DL1010	true
11	O111	First-time discount	PayPal	65	111111113	+96189012345	2024-10-11	DL1011	true
12	O112	VIP priority	Credit Card	800	654321987	+96121234567	2024-07-14	DL1012	true
13	O113	Happy Birthday!	Apple Pay	180	654321987	+96198765431	2024-03-03	DL1013	false
14	O114	Clearance sale	PayPal	450	654321987	+96134567890	2024-11-05	DL1014	false
15	O115	Loyalty customer	Cash	300	111111114	+96189012345	2024-04-22	DL1015	false

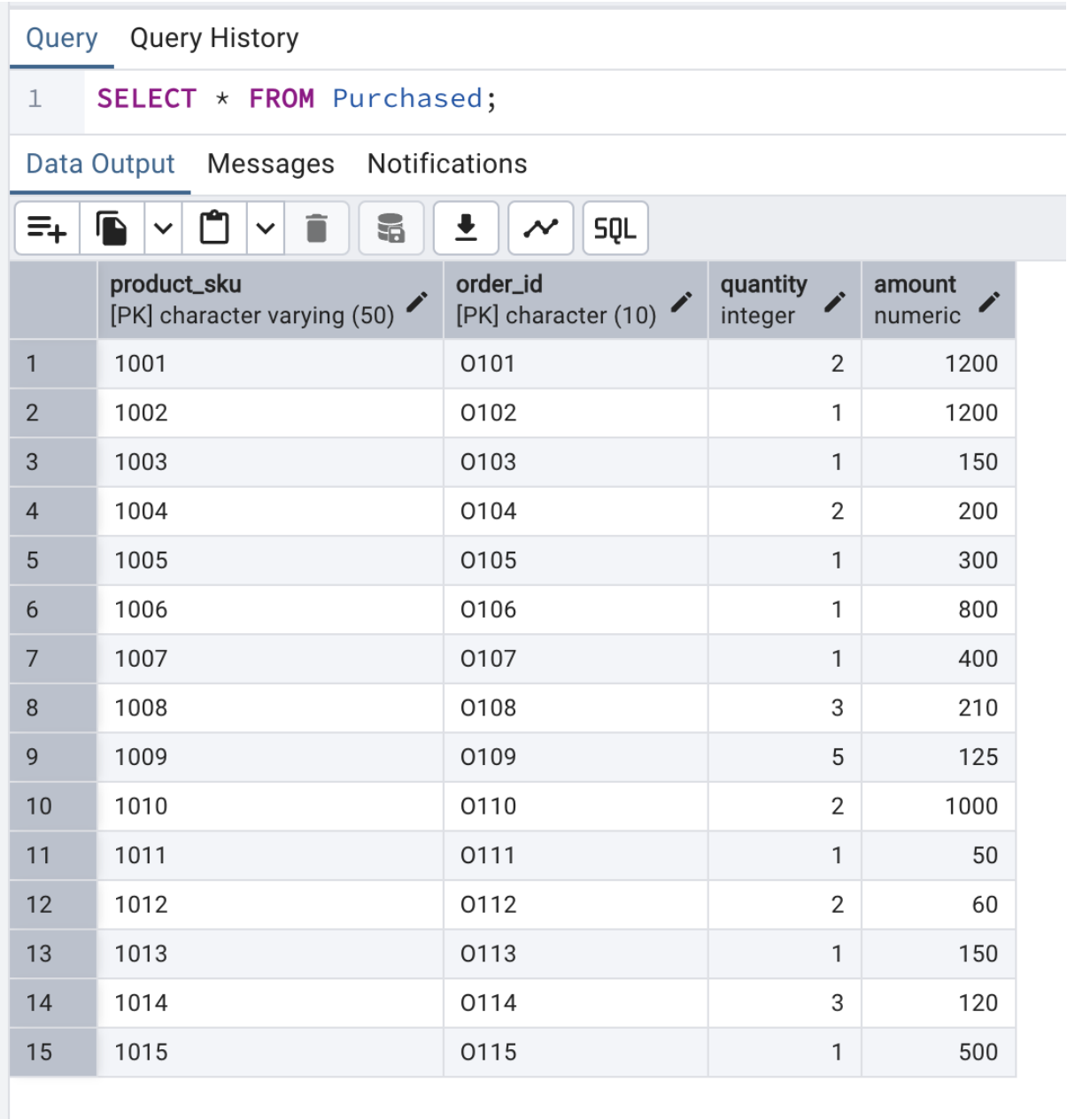
Figure 11.9: *Select from Orders Table*

11.1.4.10 Purchased

The following script selects data from the Purchased table.

```sql
SELECT * FROM Purchased;
```

Code Snippet 11.58 : *Select from Purchased Table*



	product_sku [PK] character varying (50)	order_id [PK] character (10)	quantity integer	amount numeric
1	1001	O101	2	1200
2	1002	O102	1	1200
3	1003	O103	1	150
4	1004	O104	2	200
5	1005	O105	1	300
6	1006	O106	1	800
7	1007	O107	1	400
8	1008	O108	3	210
9	1009	O109	5	125
10	1010	O110	2	1000
11	1011	O111	1	50
12	1012	O112	2	60
13	1013	O113	1	150
14	1014	O114	3	120
15	1015	O115	1	500

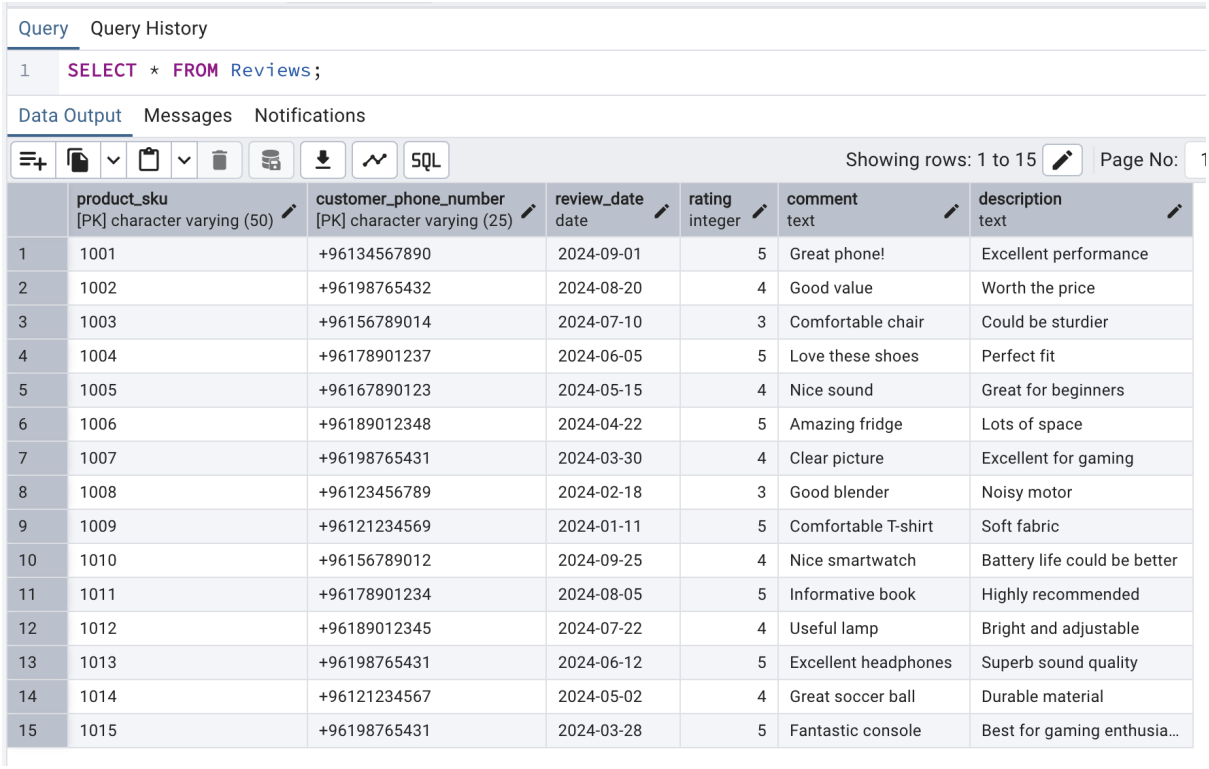
Figure 11.10: *Select from Purchased Table*

11.1.4.11 Reviews

The following script selects data from the Reviews table.

```
SELECT * FROM Reviews;
```

Code Snippet 11.59 : *Select from Reviews Table*



Query Query History

```
1  SELECT * FROM Reviews;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No:

	product_sku [PK] character varying (50)	customer_phone_number [PK] character varying (25)	review_date date	rating integer	comment text	description text
1	1001	+96134567890	2024-09-01	5	Great phone!	Excellent performance
2	1002	+96198765432	2024-08-20	4	Good value	Worth the price
3	1003	+96156789014	2024-07-10	3	Comfortable chair	Could be sturdier
4	1004	+96178901237	2024-06-05	5	Love these shoes	Perfect fit
5	1005	+96167890123	2024-05-15	4	Nice sound	Great for beginners
6	1006	+96189012348	2024-04-22	5	Amazing fridge	Lots of space
7	1007	+96198765431	2024-03-30	4	Clear picture	Excellent for gaming
8	1008	+96123456789	2024-02-18	3	Good blender	Noisy motor
9	1009	+96121234569	2024-01-11	5	Comfortable T-shirt	Soft fabric
10	1010	+96156789012	2024-09-25	4	Nice smartwatch	Battery life could be better
11	1011	+96178901234	2024-08-05	5	Informative book	Highly recommended
12	1012	+96189012345	2024-07-22	4	Useful lamp	Bright and adjustable
13	1013	+96198765431	2024-06-12	5	Excellent headphones	Superb sound quality
14	1014	+96121234567	2024-05-02	4	Great soccer ball	Durable material
15	1015	+96198765431	2024-03-28	5	Fantastic console	Best for gaming enthusia...

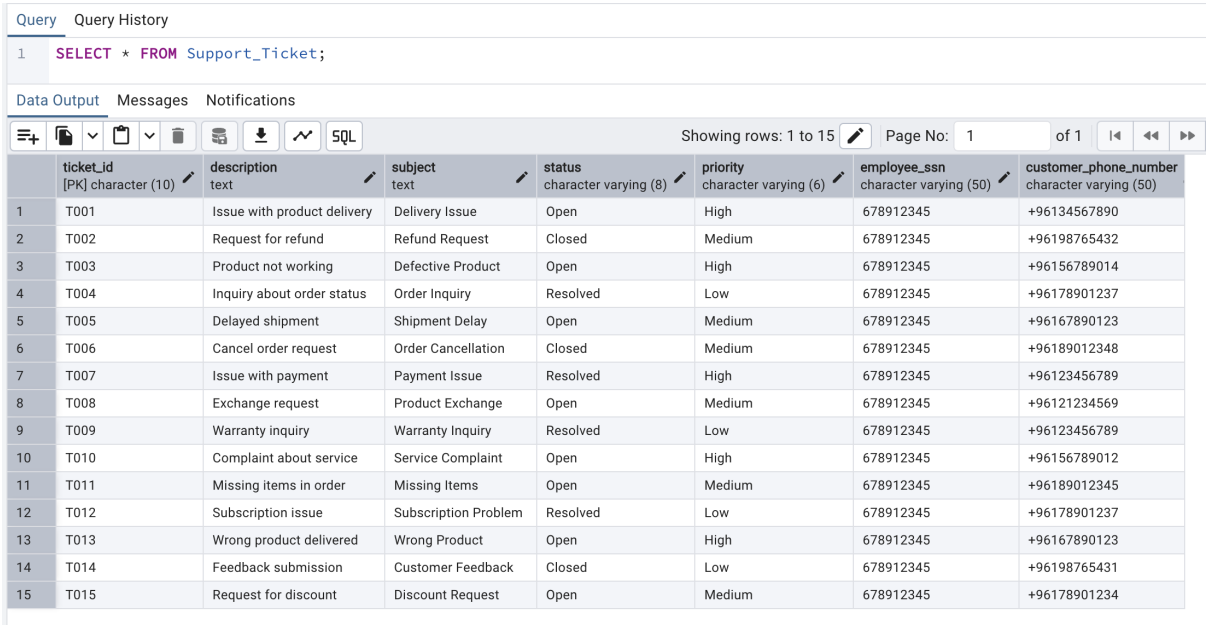
Figure 11.11: *Select from Reviews Table*

11.1.4.12 Support Ticket

The following script selects data from the Support Ticket table.

```
1  SELECT * FROM Support_Ticket;
```

Code Snippet 11.60 : *Select from Support Ticket Table*



Query Query History

```
1  SELECT * FROM Support_Ticket;
```

Data Output Messages Notifications

Showing rows: 1 to 15 Page No: 1 of 1

	ticket_id [PK] character (10)	description text	subject text	status character varying (8)	priority character varying (6)	employee_ssn character varying (50)	customer_phone_number character varying (50)
1	T001	Issue with product delivery	Delivery Issue	Open	High	678912345	+96134567890
2	T002	Request for refund	Refund Request	Closed	Medium	678912345	+96198765432
3	T003	Product not working	Defective Product	Open	High	678912345	+96156789014
4	T004	Inquiry about order status	Order Inquiry	Resolved	Low	678912345	+96178901237
5	T005	Delayed shipment	Shipment Delay	Open	Medium	678912345	+96167890123
6	T006	Cancel order request	Order Cancellation	Closed	Medium	678912345	+96189012348
7	T007	Issue with payment	Payment Issue	Resolved	High	678912345	+96123456789
8	T008	Exchange request	Product Exchange	Open	Medium	678912345	+96121234569
9	T009	Warranty inquiry	Warranty Inquiry	Resolved	Low	678912345	+96123456789
10	T010	Complaint about service	Service Complaint	Open	High	678912345	+96156789012
11	T011	Missing items in order	Missing Items	Open	Medium	678912345	+96189012345
12	T012	Subscription issue	Subscription Problem	Resolved	Low	678912345	+96178901237
13	T013	Wrong product delivered	Wrong Product	Open	High	678912345	+96167890123
14	T014	Feedback submission	Customer Feedback	Closed	Low	678912345	+96198765431
15	T015	Request for discount	Discount Request	Open	Medium	678912345	+96178901234

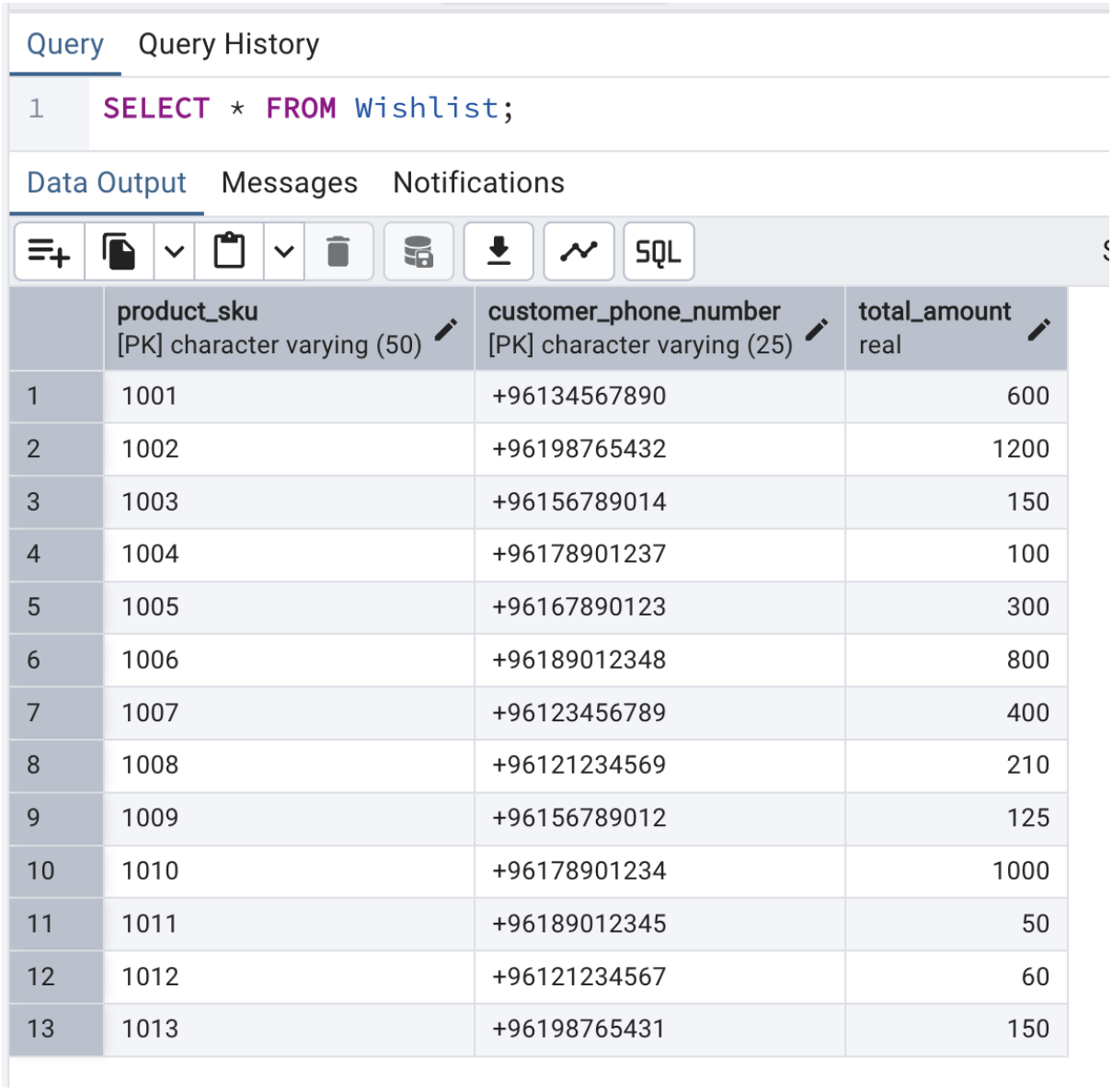
Figure 11.12: *Select from Support Ticket Table*

11.1.4.13 Wishlist

The following script selects data from the Wishlist table.

```
SELECT * FROM Wishlist;
```

Code Snippet 11.61 : *Select from Wishlist Table*



	product_sku [PK] character varying (50)	customer_phone_number [PK] character varying (25)	total_amount real
1	1001	+96134567890	600
2	1002	+96198765432	1200
3	1003	+96156789014	150
4	1004	+96178901237	100
5	1005	+96167890123	300
6	1006	+96189012348	800
7	1007	+96123456789	400
8	1008	+96121234569	210
9	1009	+96156789012	125
10	1010	+96178901234	1000
11	1011	+96189012345	50
12	1012	+96121234567	60
13	1013	+96198765431	150

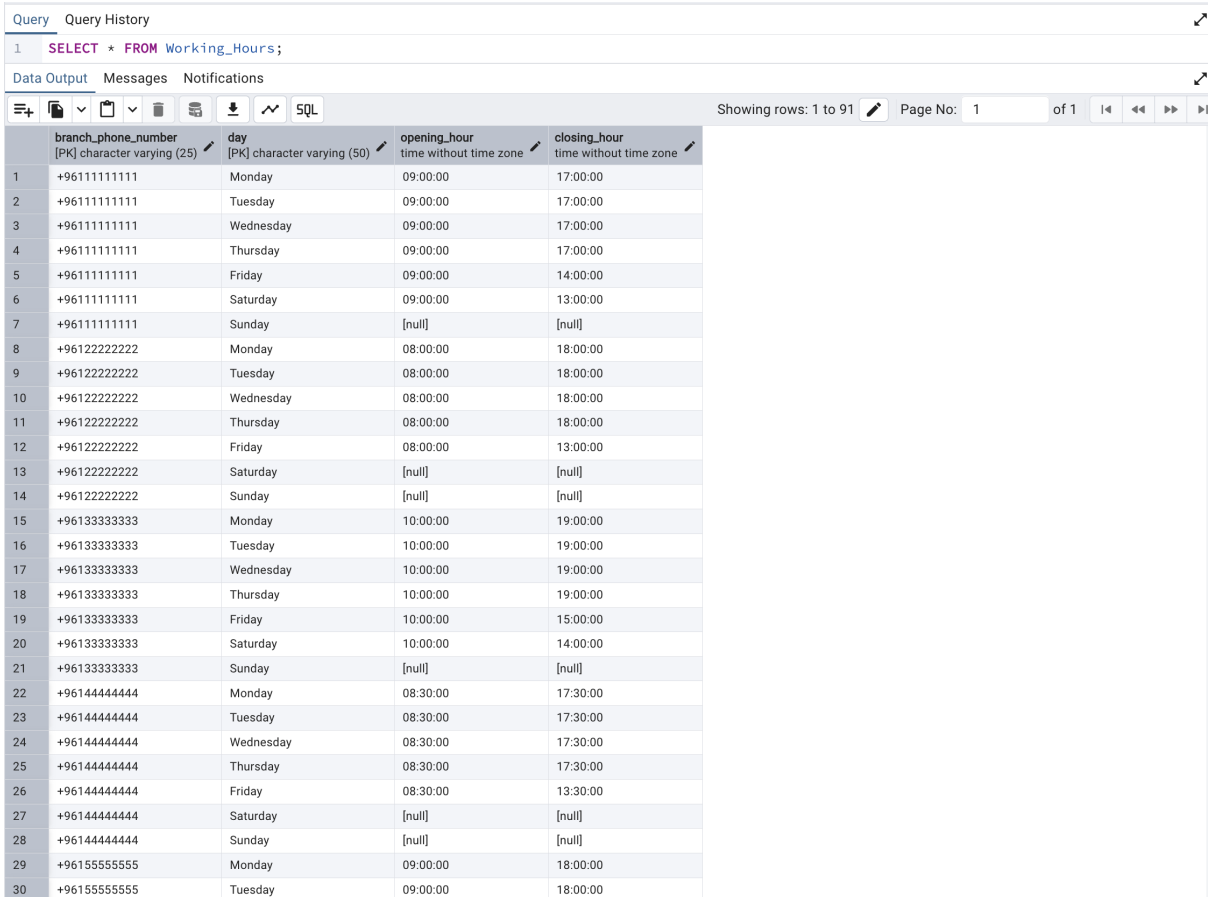
Figure 11.13: *Select from Wishlist Table*

11.1.4.14 Working Hours

The following script selects data from the Working Hours table.

```
SELECT * FROM Working_Hours;
```

Code Snippet 11.62 : *Select from Working Hours Table*



Query Query History

```
SELECT * FROM Working_Hours;
```

Data Output Messages Notifications

Showing rows: 1 to 91 Page No: 1 of 1

	branch_phone_number [PK] character varying (25)	day [PK] character varying (50)	opening_hour time without time zone	closing_hour time without time zone
1	+96111111111	Monday	09:00:00	17:00:00
2	+96111111111	Tuesday	09:00:00	17:00:00
3	+96111111111	Wednesday	09:00:00	17:00:00
4	+96111111111	Thursday	09:00:00	17:00:00
5	+96111111111	Friday	09:00:00	14:00:00
6	+96111111111	Saturday	09:00:00	13:00:00
7	+96111111111	Sunday	[null]	[null]
8	+96122222222	Monday	08:00:00	18:00:00
9	+96122222222	Tuesday	08:00:00	18:00:00
10	+96122222222	Wednesday	08:00:00	18:00:00
11	+96122222222	Thursday	08:00:00	18:00:00
12	+96122222222	Friday	08:00:00	13:00:00
13	+96122222222	Saturday	[null]	[null]
14	+96122222222	Sunday	[null]	[null]
15	+96133333333	Monday	10:00:00	19:00:00
16	+96133333333	Tuesday	10:00:00	19:00:00
17	+96133333333	Wednesday	10:00:00	19:00:00
18	+96133333333	Thursday	10:00:00	19:00:00
19	+96133333333	Friday	10:00:00	15:00:00
20	+96133333333	Saturday	10:00:00	14:00:00
21	+96133333333	Sunday	[null]	[null]
22	+96144444444	Monday	08:30:00	17:30:00
23	+96144444444	Tuesday	08:30:00	17:30:00
24	+96144444444	Wednesday	08:30:00	17:30:00
25	+96144444444	Thursday	08:30:00	17:30:00
26	+96144444444	Friday	08:30:00	13:30:00
27	+96144444444	Saturday	[null]	[null]
28	+96144444444	Sunday	[null]	[null]
29	+96155555555	Monday	09:00:00	18:00:00
30	+96155555555	Tuesday	09:00:00	18:00:00

Figure 11.14: *Select from Working Hours Table*

11.1.4.15 Dependent

The following script selects data from the Dependent table.

```
SELECT * FROM Dependent;
```

Code Snippet 11.63 : *Select from Dependent Table*

Query | Query History

```
1  SELECT * FROM Dependent;
```

Data Output | Messages | Notifications

Showing rows: 1 to 15

	employee_ssn [PK] character varying (50)	name [PK] character varying (50)	gender character varying (6)	date_of_birth date	relationship character varying (50)
1	123456789	Sarah Doe	Female	2015-04-15	Daughter
2	987654321	James Smith	Male	2013-07-20	Son
3	333333333	Emily Brown	Female	2017-02-28	Daughter
4	333333333	Lucas Jones	Male	2018-11-05	Son
5	444444444	Olivia Evans	Female	2016-09-12	Daughter
6	543216789	Ethan White	Male	2020-03-22	Son
7	666666666	Chloe Lee	Female	2014-08-01	Daughter
8	876543219	Liam Harris	Male	2015-12-15	Son
9	345678912	Mia Taylor	Female	2018-10-03	Daughter
10	888888881	Noah Wilson	Male	2021-06-08	Son
11	888888882	Sophia Martin	Female	2019-05-25	Daughter
12	789123456	Benjamin Scott	Male	2012-01-19	Son
13	999999991	Emma Anderson	Female	2015-07-13	Daughter
14	999999992	Mason Miller	Male	2018-03-09	Son
15	999999992	Ava Thomas	Female	2016-02-04	Daughter

Figure 11.15: *Select from Dependent Table*

11.1.4.16 Product Image URLs

The following script selects data from the Product Image URLs table.

```
1  SELECT * FROM Product_Image_URLs;
```

Code Snippet 11.64 : *Select from Product Image URLs Table*

Query Query History

```
1  SELECT * FROM Product_Image_URLs;
```

Data Output Messages Notifications

	product_sku [PK] character varying (50)	product_image_url [PK] character varying (50)
1	1001	https://nextstore.io/images/product/1.jpg
2	1002	https://nextstore.io/images/product/2.jpg
3	1003	https://nextstore.io/images/product/3.jpg
4	1004	https://nextstore.io/images/product/4.jpg
5	1005	https://nextstore.io/images/product/5.jpg
6	1006	https://nextstore.io/images/product/6.jpg
7	1007	https://nextstore.io/images/product/7.jpg
8	1008	https://nextstore.io/images/product/8.jpg
9	1009	https://nextstore.io/images/product/9.jpg
10	1010	https://nextstore.io/images/product/10.jpg
11	1011	https://nextstore.io/images/product/11.jpg
12	1012	https://nextstore.io/images/product/12.jpg
13	1013	https://nextstore.io/images/product/13.jpg
14	1014	https://nextstore.io/images/product/14.jpg
15	1015	https://nextstore.io/images/product/15.jpg

Figure 11.16: *Select from Product Image URLs Table*

11.1.4.17 Review Image URLs

The following script selects data from the Review Image URLs table.

```
1 SELECT * FROM Review_Image_URLs;
```

Code Snippet 11.65 : *Select from Review Image URLs Table*

Query Query History

```
1  SELECT * FROM Review_Image_URLs;
```

Data Output Messages Notifications

	product_sku [PK] character varying (50)	customer_phone_number character varying (25)	product_image_url [PK] character varying (50)
1	1001	+96134567890	https://nextstore.io/images/image1.jpg
2	1002	+96198765432	https://nextstore.io/images/image2.jpg
3	1003	+96156789014	https://nextstore.io/images/image3.jpg
4	1004	+96178901237	https://nextstore.io/images/image4.jpg
5	1005	+96167890123	https://nextstore.io/images/image5.jpg
6	1006	+96189012348	https://nextstore.io/images/image6.jpg
7	1007	+96123456789	https://nextstore.io/images/image7.jpg
8	1008	+96121234569	https://nextstore.io/images/image8.jpg
9	1009	+96156789012	https://nextstore.io/images/image9.jpg
10	1010	+96178901234	https://nextstore.io/images/image10.jpg
11	1011	+96189012345	https://nextstore.io/images/image11.jpg
12	1012	+96121234567	https://nextstore.io/images/image12.jpg
13	1013	+96198765431	https://nextstore.io/images/image13.jpg
14	1014	+96198765432	https://nextstore.io/images/image14.jpg
15	1015	+96123456789	https://nextstore.io/images/image15.jpg

Figure 11.17: *Select from Review Image URLs Table*

11.1.4.18 Located In

The following script selects data from the Located In table.

```
1  SELECT * FROM Located_in;
```

Code Snippet 11.66 : *Select from Located In Table*

Query Query History

```
1  SELECT * FROM Located_in;
```

Data Output Messages Notifications

	product_sku [PK] character varying (50)	branch_phone_number [PK] character varying (25)	quantity integer	shelf_location character varying (50)
1	1001	+96111111111	50	A1
2	1002	+96122222222	30	B2
3	1003	+96133333333	20	C3
4	1004	+96144444444	15	D4
5	1005	+96155555555	60	E5
6	1006	+96166666666	25	F6
7	1007	+96177777777	10	G7
8	1008	+96188888888	35	H8
9	1009	+96199999999	40	I9
10	1010	+96112345678	50	J10
11	1011	+96124681357	70	K11
12	1012	+96124681357	20	L12
13	1013	+96165432198	15	M13
14	1014	+96165432198	5	N14
15	1015	+96111223344	55	O15

Figure 11.18: *Select from Located In Table*

11.1.4.19 Colors

The following script selects data from the Colors table.

```
1  SELECT * FROM Colors;
```

Code Snippet 11.67 : *Select from Colors Table*

Query Query History

```
1  SELECT * FROM Colors;
```

Data Output Messages Notifications

	product_sku [PK] character varying (50)	product_color [PK] character varying (50)
1	1001	Red
2	1002	Blue
3	1003	Green
4	1004	Black
5	1005	White
6	1006	Yellow
7	1007	Pink
8	1008	Purple
9	1009	Orange
10	1010	Brown
11	1011	Gray
12	1012	Gold
13	1013	Silver
14	1014	Beige
15	1015	Navy

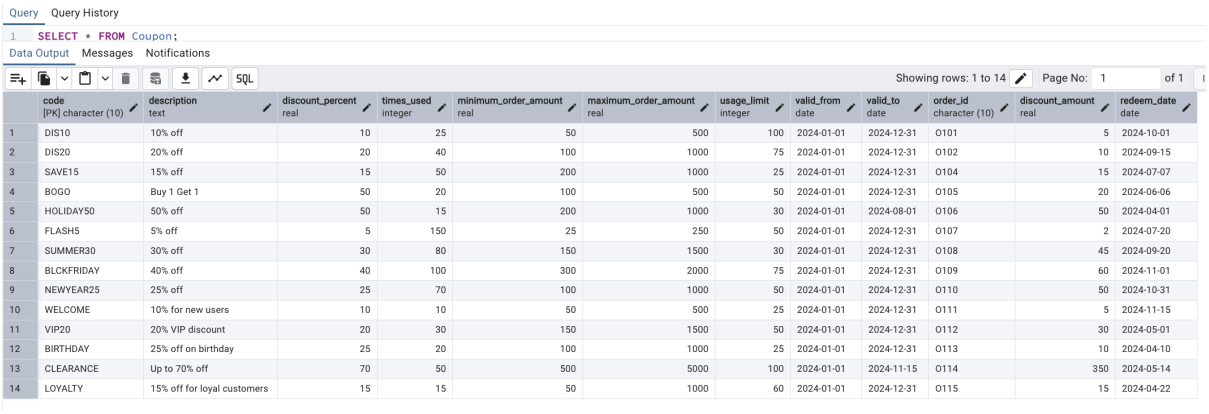
Figure 11.19: *Select from Colors Table*

11.1.4.20 Coupon

The following script selects data from the Coupon table.

```sql
SELECT * FROM Coupon;
```

Code Snippet 11.68 : *Select from Coupon Table*



Query Query History

```
SELECT * FROM Coupon;
```

Data Output Messages Notifications

Showing rows: 1 to 14 Page No: 1 of 1

	code [PK] character (10)	description text	discount_percent real	times_used integer	minimum_order_amount real	maximum_order_amount real	usage_limit integer	valid_from date	valid_to date	order_id character (10)	discount_amount real	redeem_date date
1	DIS10	10% off	10	25	50	500	100	2024-01-01	2024-12-31	O101	5	2024-10-01
2	DIS20	20% off	20	40	100	1000	75	2024-01-01	2024-12-31	O102	10	2024-09-15
3	SAVE15	15% off	15	50	200	1000	25	2024-01-01	2024-12-31	O104	15	2024-07-07
4	BOGO	Buy 1 Get 1	50	20	100	500	50	2024-01-01	2024-12-31	O105	20	2024-06-06
5	HOLIDAY50	50% off	50	15	200	1000	30	2024-01-01	2024-08-01	O106	50	2024-04-01
6	FLASH5	5% off	5	150	25	250	50	2024-01-01	2024-12-31	O107	2	2024-07-20
7	SUMMER30	30% off	30	80	150	1500	30	2024-01-01	2024-12-31	O108	45	2024-09-20
8	BLCKFRIDAY	40% off	40	100	300	2000	75	2024-01-01	2024-12-31	O109	60	2024-11-01
9	NEWYEAR25	25% off	25	70	100	1000	50	2024-01-01	2024-12-31	O110	50	2024-10-31
10	WELCOME	10% for new users	10	10	50	500	25	2024-01-01	2024-12-31	O111	5	2024-11-15
11	VIP20	20% VIP discount	20	30	150	1500	50	2024-01-01	2024-12-31	O112	30	2024-05-01
12	BIRTHDAY	25% off on birthday	25	20	100	1000	25	2024-01-01	2024-12-31	O113	10	2024-04-10
13	CLEARANCE	Up to 70% off	70	50	500	5000	100	2024-01-01	2024-11-15	O114	350	2024-05-14
14	LOYALTY	15% off for loyal customers	15	15	50	1000	60	2024-01-01	2024-12-31	O115	15	2024-04-22

Figure 11.20: *Select from Coupon Table*

11.1.4.21 Department Location

The following script selects data from the Department Location table.

```sql
SELECT * FROM Department_Location;
```

Code Snippet 11.69 : *Select from Department Location Table*

Query Query History

```
1  SELECT * FROM Department_Location;
```

Data Output Messages Notifications

SQL

	department_name [PK] character varying (50)	location [PK] character varying (50)
1	Sales	Beirut
2	Marketing	Tripoli
3	HR	Zahle
4	Finance	Jounieh
5	Operations	Sidon
6	IT	Byblos
7	Customer Support	Tyre
8	Logistics	Baalbek
9	Legal	Batroun
10	Training	Dora
11	Facilities	Antelias
12	R&D	Achrafieh

Figure 11.21: *Select from Department Location Table*

11.1.5 Complex Queries

11.1.5.1 Customer with Highest Number of Orders

```
SELECT
  Customer.First_name,
  Customer.Last_name,
  COUNT(Orders.Order_id) AS Total_Orders
FROM
  Customer
```

```
  JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
GROUP BY
  Customer.First_name,
  Customer.Last_name
ORDER BY
  Total_Orders DESC
FETCH FIRST
  5 ROWS ONLY;
```

Code Snippet 11.70 : *Customer with Highest Number of Orders*

Query | Query History

```
SELECT
  Customer.First_name,
  Customer.Last_name,
  COUNT(Orders.Order_id) AS Total_Orders
FROM
  Customer
  JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
GROUP BY
  Customer.First_name,
  Customer.Last_name
ORDER BY
  Total_Orders DESC
FETCH FIRST
  5 ROWS ONLY;
```

Data Output | Messages | Notifications

	first_name character varying (50)	last_name character varying (50)	total_orders bigint
1	Isabella	Taylor	2
2	John	Doe	2
3	Diana	Lee	1
4	Emma	White	1
5	Bob	Jones	1

Figure 11.22: *Customer with Highest Number of Orders*

This SQL query retrieves the top 5 customers who have placed the highest number of orders. It joins the Customer and Order tables on the Phone_number and Customer_phone_number fields, respectively. The query groups the results by the customers' first and last names, counts the number of orders for each customer, and orders the results in descending order based on the total number of orders. Finally, it limits the output to the top 5 customers. This query is useful for identifying the most active customers, which can help in targeting marketing efforts, improving customer service, and understanding customer behavior.

11.1.5.2 Products with Highest Revenue

```
SELECT
  Product.Name AS Product_Name,
  SUM(Purchased.Quantity) AS Total_Quantity_Sold,
  SUM(Purchased.Amount) AS Total_Revenue_Generated
FROM
  Product,
  Purchased
WHERE
  Product.SKU = Purchased.Product_SKU
GROUP BY
  Product.Name
ORDER BY
  Total_Revenue_Generated DESC
FETCH FIRST
  5 ROWS ONLY;
```

Code Snippet 11.71 : *Products with Highest Revenue*

Query | Query History

```sql
SELECT
  Product.Name AS Product_Name,
  SUM(Purchased.Quantity) AS Total_Quantity_Sold,
  SUM(Purchased.Amount) AS Total_Revenue_Generated
FROM
  Product,
  Purchased
WHERE
  Product.SKU = Purchased.Product_SKU
GROUP BY
  Product.Name
ORDER BY
  Total_Revenue_Generated DESC
FETCH FIRST
  5 ROWS ONLY;
```

Data Output | Messages | Notifications

	product_name character varying (50)	total_quantity_sold bigint	total_revenue_generated numeric
1	Smartphone	2	1200
2	Laptop	1	1200
3	Watch	2	1000
4	Refrigerator	1	800
5	Gaming Console	1	500

Figure 11.23: *Products with Highest Revenue*

This SQL query retrieves the top 5 products that have generated the highest revenue. It selects the product name, the total quantity sold, and the total revenue generated for each product. The query joins the Product and Purchased tables on the SKU and Product_SKU columns, respectively. It then groups the results by product name and orders them by total revenue in descending order. Finally, it limits the results to the top 5 rows. This is useful for businesses to identify their best-selling products and make informed decisions about inventory, marketing, and sales strategies.

11.1.5.3 Most Sold Product in Each Branch

```sql
WITH
  Product_Sales AS (
```

```
    SELECT
      L.Branch_phone_number,
      P.Product_SKU,
      SUM(P.Quantity) AS Total_Sold
    FROM
      Purchased P
      JOIN Located_in L ON P.Product_SKU = L.Product_SKU
    GROUP BY
      L.Branch_phone_number,
      P.Product_SKU
  ),
  Max_Sales_Per_Branch AS (
    SELECT
      Branch_phone_number,
      MAX(Total_Sold) AS Max_Sold
    FROM
      Product_Sales
    GROUP BY
      Branch_phone_number
  )
SELECT
  PS.Branch_phone_number,
  PS.Product_SKU,
  PS.Total_Sold
FROM
  Product_Sales PS
  JOIN Max_Sales_Per_Branch MSPB ON PS.Branch_phone_number = MSPB.
      Branch_phone_number
  AND PS.Total_Sold = MSPB.Max_Sold;
```

Code Snippet 11.72 : *Most Sold Product in Each Branch*

Query Query History

```sql
WITH Product_Sales AS (
    SELECT
        L.Branch_phone_number,
        P.Product_SKU,
        SUM(P.Quantity) AS Total_Sold
    FROM
        Purchased P
    JOIN
        Located_in L ON P.Product_SKU = L.Product_SKU
    GROUP BY
        L.Branch_phone_number, P.Product_SKU
),
Max_Sales_Per_Branch AS (
    SELECT
        Branch_phone_number,
        MAX(Total_Sold) AS Max_Sold
    FROM
        Product_Sales
    GROUP BY
        Branch_phone_number
)
SELECT
    PS.Branch_phone_number,
    PS.Product_SKU,
    PS.Total_Sold
FROM
    Product_Sales PS
JOIN
    Max_Sales_Per_Branch MSPB
    ON PS.Branch_phone_number = MSPB.Branch_phone_number
    AND PS.Total_Sold = MSPB.Max_Sold;
```

Data Output Messages Notifications

	branch_phone_number character varying (25)	product_sku character varying (50)	total_sold bigint
1	+96112345678	1010	2
2	+96155555555	1005	1
3	+96124681357	1012	2
4	+96166666666	1006	1
5	+96188888888	1008	3
6	+96122222222	1002	1
7	+96111111111	1001	2
8	+96133333333	1003	1
9	+96199999999	1009	5
10	+96177777777	1007	1
11	+96144444444	1004	2
12	+96111223344	1015	1
13	+96165432198	1014	3

Figure 11.24: *Most Sold Product in Each Branch*

This SQL query identifies the most sold product in each branch by calculating the total quantity sold for each product at each branch. It first joins the Purchased and Located_in tables on the Product_SKU

field. Then, it groups the results by branch phone number and product SKU, summing the quantities sold. The HAVING clause filters these groups to only include the product with the maximum total quantity sold per branch. This is useful for businesses to understand which products are the top sellers at each branch, enabling better inventory management and targeted marketing strategies.

11.1.5.4 Calculate Customer Lifetime Value for Each Customer

```
SELECT
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name,
  SUM(Purchased.Amount) AS Lifetime_Value
FROM
  Customer
  JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
  JOIN Purchased ON Orders.Order_id = Purchased.Order_id
GROUP BY
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name
ORDER BY
  Lifetime_Value DESC;
```

Code Snippet 11.73 : *Calculate Customer Lifetime Value for Each Customer*

Query | Query History

```sql
SELECT
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name,
  SUM(Purchased.Amount) AS Lifetime_Value
FROM
  Customer
  JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
  JOIN Purchased ON Orders.Order_id = Purchased.Order_id
GROUP BY
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name
ORDER BY
  Lifetime_Value DESC;
```

Data Output | Messages | Notifications

	phone_number [PK] character varying (50)	first_name character varying (50)	last_name character varying (50)	lifetime_value numeric
1	+96134567890	John	Doe	1320
2	+96198765432	Jane	Smith	1200
3	+96178901234	Jack	Miller	1000
4	+96189012348	Diana	Lee	800
5	+96189012345	Isabella	Taylor	550
6	+96123456789	Frank	Wilson	400
7	+96167890123	Charlie	Evans	300
8	+96121234569	Emma	White	210
9	+96178901237	Bob	Jones	200
10	+96156789014	Alice	Brown	150
11	+96198765431	Mike	Anderson	150
12	+96156789012	George	King	125
13	+96121234567	Kevin	Harris	60

Figure 11.25: *Calculate Customer Lifetime Value for Each Customer*

This SQL query calculates the Customer Lifetime Value for each customer by summing up the total amount spent by each customer across all their orders. It joins three tables: Customer, Orders, and Purchased, using the customer's phone number and order ID to link the data. The results are grouped by the customer's phone number, first name, and last name, and then ordered by the lifetime value in descending order. This is useful for businesses to identify their most valuable customers, enabling them to tailor marketing strategies, improve customer retention, and allocate resources more effectively.

11.1.5.5 Underperforming Products

```
SELECT
  Product.SKU,
  Product.Name,
  SUM(Purchased.Quantity) AS Total_Sold,
  Product.Price * SUM(Purchased.Quantity) AS Total_Revenue,
  COUNT(DISTINCT Purchased.Order_id) AS Total_Orders
FROM
  Product
  LEFT JOIN Purchased ON Product.SKU = Purchased.Product_SKU
GROUP BY
  Product.SKU,
  Product.Name,
  Product.Price
HAVING
  SUM(Purchased.Quantity) < 10
  OR Product.Price * SUM(Purchased.Quantity) < 500;
```

Code Snippet 11.74 : *Underperforming Products*

Query | Query History

```sql
SELECT
  Product.SKU,
  Product.Name,
  SUM(Purchased.Quantity) AS Total_Sold,
  Product.Price * SUM(Purchased.Quantity) AS Total_Revenue,
  COUNT(DISTINCT Purchased.Order_id) AS Total_Orders
FROM
  Product
  LEFT JOIN Purchased ON Product.SKU = Purchased.Product_SKU
GROUP BY
  Product.SKU,
  Product.Name,
  Product.Price
HAVING
  SUM(Purchased.Quantity) < 10
  OR Product.Price * SUM(Purchased.Quantity) < 500;
```

Data Output | Messages | Notifications

	sku [PK] character varying (50)	name character varying (50)	total_sold bigint	total_revenue bigint	total_orders bigint
1	1001	Smartphone	2	1200	1
2	1002	Laptop	1	1200	1
3	1003	Office Chair	1	150	1
4	1004	Running Shoes	2	200	1
5	1005	Acoustic Guitar	1	300	1
6	1006	Refrigerator	1	800	1
7	1007	LED TV	1	400	1
8	1008	Blender	3	210	1
9	1009	T-Shirt	5	125	1
10	1010	Watch	2	1000	1
11	1011	Textbook	1	50	1
12	1012	Desk Lamp	2	60	1
13	1013	Wireless Headphones	1	150	1
14	1014	Soccer Ball	3	120	1
15	1015	Gaming Console	1	500	1

Figure 11.26: *Underperforming Products*

This SQL query identifies underperforming products by selecting products that have either sold fewer than 10 units or generated less than $500 in total revenue. It joins the Product table with the Purchased table on the SKU field to aggregate sales data. The query calculates the total quantity sold (Total_Sold), total revenue (Total_Revenue), and the number of distinct orders (Total_Orders) for each product. The HAVING clause filters the results to include only those products that meet the underperformance criteria. This is useful for businesses to identify products that may need marketing attention,

discounts, or discontinuation.

11.1.5.6 Find Popular Products Combinations

```sql
WITH
  ProductPairs AS (
    SELECT
      p1.Product_SKU AS Product_1,
      p2.Product_SKU AS Product_2,
      COUNT(*) AS Pair_Count
    FROM
      Purchased p1
      JOIN Purchased p2 ON p1.Order_id = p2.Order_id
      AND p1.Product_SKU < p2.Product_SKU
    GROUP BY
      p1.Product_SKU,
      p2.Product_SKU
  )
SELECT
  Product_1,
  Product_2,
  Pair_Count
FROM
  ProductPairs
WHERE
  Pair_Count > 5
ORDER BY
  Pair_Count DESC;
```

Code Snippet 11.75 : *Find Popular Products Combinations*

Query Query History

```sql
WITH
  ProductPairs AS (
    SELECT
      p1.Product_SKU AS Product_1,
      p2.Product_SKU AS Product_2,
      COUNT(*) AS Pair_Count
    FROM
      Purchased p1
      JOIN Purchased p2 ON p1.Order_id = p2.Order_id
      AND p1.Product_SKU < p2.Product_SKU
    GROUP BY
      p1.Product_SKU,
      p2.Product_SKU
  )
SELECT
  Product_1,
  Product_2,
  Pair_Count
FROM
  ProductPairs
WHERE
  Pair_Count > 5
ORDER BY
  Pair_Count DESC;
```

Data Output Messages Notifications

	product_1 character varying (50)	product_2 character varying (50)	pair_count bigint

Figure 11.27: *Find Popular Products Combinations*

This SQL query identifies popular product combinations that are frequently purchased together. It uses a Common Table Expression (CTE) named ProductPairs to join the Purchased table with itself, matching rows where the Order_id is the same but the Product_SKU is different. The condition p1.Product_SKU ¡ p2.Product_SKU ensures that each pair is counted only once. The query then groups these pairs and counts how often each combination appears. Finally, it selects pairs that appear more than five times and orders them by their frequency in descending order. This information is useful for un-

derstanding customer purchasing behavior, which can inform marketing strategies, product placement, and inventory management.

11.1.5.7 Products with the Most Discounts

```
SELECT
  Coupon.Code,
  Product.SKU,
  Product.Name,
  COUNT(*) AS Times_Discounted,
  SUM(Coupon.Discount_amount) AS Total_Discount_Amount
FROM
  Coupon
  JOIN Orders ON Coupon.Order_ID = Orders.Order_ID
  JOIN Purchased ON Orders.Order_ID = Purchased.Order_ID
  JOIN Product ON Purchased.Product_SKU = Product.SKU
GROUP BY
  Coupon.Code,
  Product.SKU,
  Product.Name
ORDER BY
  Total_Discount_Amount DESC;
```

Code Snippet 11.76 : *Products with the Most Discounts*

Query | Query History

```
1  SELECT
2    Coupon.Code,
3    Product.SKU,
4    Product.Name,
5    COUNT(*) AS Times_Discounted,
6    SUM(Coupon.Discount_amount) AS Total_Discount_Amount
7  FROM
8    Coupon
9    JOIN Orders ON Coupon.Order_ID = Orders.Order_ID
10   JOIN Purchased ON Orders.Order_ID = Purchased.Order_ID
11   JOIN Product ON Purchased.Product_SKU = Product.SKU
12 GROUP BY
13   Coupon.Code,
14   Product.SKU,
15   Product.Name
16 ORDER BY
17   Total_Discount_Amount DESC;
```

Data Output | Messages | Notifications

SQL

	code character (10)	sku character varying (50)	name character varying (50)	times_discounted bigint	total_discount_amount real
1	CLEARANCE	1014	Soccer Ball	1	350
2	BLCKFRIDAY	1009	T-Shirt	1	60
3	HOLIDAY50	1006	Refrigerator	1	50
4	NEWYEAR25	1010	Watch	1	50
5	SUMMER30	1008	Blender	1	45
6	VIP20	1012	Desk Lamp	1	30
7	BOGO	1005	Acoustic Guitar	1	20
8	SAVE15	1004	Running Shoes	1	15
9	LOYALTY	1015	Gaming Console	1	15
10	DIS20	1002	Laptop	1	10
11	BIRTHDAY	1013	Wireless Headphones	1	10
12	WELCOME	1011	Textbook	1	5
13	DIS10	1001	Smartphone	1	5
14	FLASH5	1007	LED TV	1	2

Figure 11.28: *Products with the Most Discounts*

This SQL query retrieves and aggregates data about discounts applied to products. It joins four tables: Coupon, Orders, Purchased, and Product. The query selects the coupon code, product SKU, product name, the count of times each coupon was used (Times_Discounted), and the total discount amount for each product (Total_Discount_Amount). The results are grouped by coupon code, product SKU, and product name, and then ordered by the total discount amount in descending order. This is useful for analyzing the effectiveness of different coupons and understanding which products benefit most from discounts, aiding in marketing and sales strategy decisions.

11.1.5.8 Seasonal Trends for Product Categories

```sql
SELECT
  EXTRACT(
    MONTH
    FROM
      Orders.Date
  ) AS Month,
  Product.Category_name,
  SUM(Purchased.Quantity) AS Total_Sold
FROM
  Orders
  JOIN Purchased ON Orders.Order_id = Purchased.Order_id
  JOIN Product ON Purchased.Product_SKU = Product.SKU
GROUP BY
  EXTRACT(
    MONTH
    FROM
      Orders.Date
  ),
  Product.Category_name
ORDER BY
  Month,
  Total_Sold DESC;
```

Code Snippet 11.77 : *Seasonal Trends for Product Categories*

Query Query History

```
SELECT
  EXTRACT(
    MONTH
    FROM
      Orders.Date
  ) AS Month,
  Product.Category_name,
  SUM(Purchased.Quantity) AS Total_Sold
FROM
  Orders
  JOIN Purchased ON Orders.Order_id = Purchased.Order_id
  JOIN Product ON Purchased.Product_SKU = Product.SKU
GROUP BY
  EXTRACT(
    MONTH
    FROM
      Orders.Date
  ),
  Product.Category_name
ORDER BY
  Month,
  Total_Sold DESC;
```

Data Output Messages Notifications

SQL

	month numeric	category_name character varying (50)	total_sold bigint
1	2	Jewelry	2
2	3	Electronics	1
3	4	Sports	2
4	4	Electronics	1
5	6	Electronics	3
6	7	Furniture	3
7	7	Electronics	1
8	9	Electronics	1
9	10	Electronics	2
10	10	Books	1
11	11	Clothing	5
12	11	Sports	3
13	11	Electronics	1
14	11	Music	1

Figure 11.29: *Seasonal Trends for Product Categories*

This SQL query is designed to analyze seasonal trends in product categories by aggregating sales data on a monthly basis. It extracts the month from the Orders.Date field and groups the results by both the month and the product category name. The query then sums the quantity of products sold (Purchased.Quantity) for each category within each month. The results are ordered first by month and then by the total quantity sold in descending order. This analysis is useful for identifying which product categories perform best during different times of the year, enabling businesses to make informed decisions about inventory management, marketing strategies, and sales forecasting.

11.1.5.9 Branches with Stock Shortages

```sql
WITH
  StockLevels AS (
    SELECT
      Branch_phone_number,
      Product_SKU,
      Quantity,
      ROW_NUMBER() OVER (
        PARTITION BY
          Branch_phone_number
        ORDER BY
          Quantity ASC
      ) AS Stock_Rank
    FROM
      Located_in
  )
SELECT
  Branch.Phone_number AS Branch_ID,
  Branch.Name AS Branch_Name,
  StockLevels.Product_SKU,
  StockLevels.Quantity AS Current_Stock
FROM
  Branch
  JOIN StockLevels ON Branch.Phone_number = StockLevels.Branch_phone_number
WHERE
  StockLevels.Stock_Rank <= 3
ORDER BY
  Branch_ID,
  Current_Stock ASC;
```

Code Snippet 11.78 : *Branches with Stock Shortages*

Query | Query History

```sql
WITH
  StockLevels AS (
    SELECT
      Branch_phone_number,
      Product_SKU,
      Quantity,
      ROW_NUMBER() OVER (
        PARTITION BY
          Branch_phone_number
        ORDER BY
          Quantity ASC
      ) AS Stock_Rank
    FROM
      Located_in
  )
SELECT
  Branch.Phone_number AS Branch_ID,
  Branch.Name AS Branch_Name,
  StockLevels.Product_SKU,
  StockLevels.Quantity AS Current_Stock
FROM
  Branch
  JOIN StockLevels ON Branch.Phone_number = StockLevels.Branch_phone_number
WHERE
  StockLevels.Stock_Rank <= 3
ORDER BY
  Branch_ID,
  Current_Stock ASC;
```

Data Output | Messages | Notifications

	branch_id character varying (25)	branch_name character varying (50)	product_sku character varying (50)	current_stock integer
1	+96111111111	Beirut Main	1001	50
2	+96111223344	Choueifat Station	1015	55
3	+96112345678	Downtown Center	1010	50
4	+96122222222	Tripoli Branch	1002	30
5	+96124681357	Dora Warehouse	1012	20
6	+96124681357	Dora Warehouse	1011	70
7	+96133333333	Sidon Hub	1003	20
8	+96144444444	Zahle Branch	1004	15
9	+96155555555	Jounieh Store	1005	60
10	+96165432198	Aley Branch	1014	5
11	+96165432198	Aley Branch	1013	15
12	+96166666666	Byblos Outlet	1006	25
13	+96177777777	Tyre Shop	1007	10
14	+96188888888	Baalbek Point	1008	35
15	+96199999999	Batroun Corner	1009	40

Figure 11.30: *Branches with Stock Shortages*

This SQL query identifies branches with stock shortages by ranking products based on their quantity at each branch. The StockLevels Common Table Expression (CTE) calculates the rank of each product's stock quantity within each branch using the ROW_NUMBER() function. The main query then selects branches and their products where the stock quantity ranks in the lowest three (Stock_Rank ¡= 3). This information is useful for the company to quickly identify and address potential stock shortages, ensuring that branches are adequately stocked and can meet customer demand, thereby improving inventory management and operational efficiency.

11.1.5.10 Best Performing Branch

```
WITH
  BranchSales AS (
    SELECT
      Branch_phone_number,
      SUM(Purchased.Amount) AS Total_Revenue
    FROM
      Purchased
      JOIN Located_in ON Purchased.Product_SKU = Located_in.Product_SKU
    GROUP BY
      Branch_phone_number
  )
SELECT
  Branch_phone_number,
  Total_Revenue
FROM
  (
    SELECT
      Branch_phone_number,
      Total_Revenue,
      RANK() OVER (
        ORDER BY
          Total_Revenue DESC
      ) AS Rank
    FROM
      BranchSales
  ) RankedSales
WHERE
  Rank = 1;
```

Code Snippet 11.79 : *Best Performing Branch*

Query Query History

```sql
WITH
  BranchSales AS (
    SELECT
      Branch_phone_number,
      SUM(Purchased.Amount) AS Total_Revenue
    FROM
      Purchased
      JOIN Located_in ON Purchased.Product_SKU = Located_in.Product_SKU
    GROUP BY
      Branch_phone_number
  )
SELECT
  Branch_phone_number,
  Total_Revenue
FROM
  (
    SELECT
      Branch_phone_number,
      Total_Revenue,
      RANK() OVER (
        ORDER BY
          Total_Revenue DESC
      ) AS Rank
    FROM
      BranchSales
  ) RankedSales
WHERE
  Rank = 1;
```

Data Output Messages Notifications

	branch_phone_number character varying (25)	total_revenue numeric
1	+96111111111	1200
2	+96122222222	1200

Figure 11.31: *Best Performing Branch*

This SQL query identifies the best-performing branch based on total revenue from sales. It first calculates the total revenue for each branch by summing the amounts from the Purchased table, joined with the Located_in table to associate products with branches. The results are grouped by branch phone number. Then, it ranks these branches by total revenue in descending order. Finally, it selects the branch with the highest total revenue (ranked 1). This information is useful for the company as it highlights the most successful branch, allowing management to analyze and replicate its strategies across other branches to improve overall performance.

11.2 Views

11.2.1 Customer orders

```
CREATE VIEW
  Customer_Orders AS
SELECT
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name,
  COUNT(Order.Order_id) AS Total_Orders
FROM
  Customer
  JOIN Order ON Customer.Phone_number = Order.Customer_phone_number
GROUP BY
  Customer.Phone_number,
  Customer.First_name,
  Customer.Last_name;

SELECT * FROM Customer_Orders;
```

Code Snippet 11.80 : *Customer Orders View*

Query Query History

```
1  CREATE VIEW
2    Customer_Orders AS
3  SELECT
4    Customer.Phone_number,
5    Customer.First_name,
6    Customer.Last_name,
7    COUNT(Orders.Order_id) AS Total_Orders
8  FROM
9    Customer
10   JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
11 GROUP BY
12   Customer.Phone_number,
13   Customer.First_name,
14   Customer.Last_name;
15
16 SELECT * FROM Customer_Orders;
```

Data Output Messages Notifications

Showing r

	phone_number character varying (50)	first_name character varying (50)	last_name character varying (50)	total_orders bigint
1	+96198765432	Jane	Smith	1
2	+96178901237	Bob	Jones	1
3	+96156789014	Alice	Brown	1
4	+96167890123	Charlie	Evans	1
5	+96189012345	Isabella	Taylor	2
6	+96121234567	Kevin	Harris	1
7	+96134567890	John	Doe	2
8	+96156789012	George	King	1
9	+96123456789	Frank	Wilson	1
10	+96178901234	Jack	Miller	1
11	+96198765431	Mike	Anderson	1
12	+96189012348	Diana	Lee	1
13	+96121234569	Emma	White	1

Figure 11.32: *Customer Orders View*

Creating the Customer_Orders view is essential for simplifying and organizing data retrieval related to customer orders. This view consolidates information from the Customer and Order tables, providing a clear and concise summary of each customer's total orders. By joining these tables on the customer's phone number and grouping the results by customer details, the view enables efficient querying and reporting without repeatedly writing complex SQL joins and aggregations. This enhances readability, maintainability, and performance of database operations, making it easier for analysts and developers to access and analyze customer order data.

11.2.2 Update of the number of employees in the Department

```
1 CREATE VIEW
```

```
  Department_Employees AS
SELECT
  Department_name,
  COUNT(Employee.SSN) AS Total_Employees
FROM
  Department
  JOIN Employee ON Department.name = Employee.Department_name
GROUP BY
  Department_name;

SELECT * FROM Department_Employees;
```

Code Snippet 11.81 : *Update of the Number of Employees in the Department View*

Query Query History

```
1  CREATE VIEW
2    Department_Employees AS
3  SELECT
4    Department_name,
5    COUNT(Employee.SSN) AS Total_Employees
6  FROM
7    Department
8    JOIN Employee ON Department.name = Employee.Department_name
9  GROUP BY
10   Department_name;
11
12 SELECT * FROM Department_Employees;
```

Data Output Messages Notifications

Showing rows: 1 t

	department_name character varying (50)	total_employees bigint
1	Logistics	18
2	Marketing	2
3	Operations	2
4	Legal	1
5	Finance	3
6	Facilities	1
7	Training	4
8	R&D	1
9	Customer Support	2
10	Sales	6
11	IT	1
12	HR	2

Figure 11.33: *Update of the Number of Employees in the Department View*

The Department_Employees view is necessary for our DBMS as it provides a simplified and aggregated representation of the number of employees in each department. By creating this view, we can easily query and retrieve the total count of employees per department without repeatedly writing complex SQL joins and group by operations. This enhances query efficiency, improves readability, and ensures consistency in how this specific data is accessed and reported across different parts of the application.

11.2.3 Total Revenue Generated by Each Product

```
1 CREATE VIEW
2   Product_Revenue AS
3 SELECT
```

```
  Product.SKU,
  Product.Name,
  SUM(Purchased.Amount) AS Total_Revenue
FROM
  Product
  JOIN Purchased ON Product.SKU = Purchased.Product_SKU
GROUP BY
  Product.SKU,
  Product.Name;

SELECT * FROM Product_Revenue;
```

Code Snippet 11.82 : *Total Revenue Generated by Each Product View*

Creating the Product_Revenue view is necessary to simplify and streamline the process of analyzing the total revenue generated by each product. By encapsulating the SQL query within a view, you provide a reusable and easily accessible way to retrieve this aggregated data without repeatedly writing the same complex query. This enhances code maintainability, improves readability, and ensures consistency in how revenue data is calculated and presented across different parts of the application or for various reporting purposes

11.3 Triggers

11.3.1 Update Product Revenue Trigger

```
CREATE OR REPLACE FUNCTION Update_Product_Revenue()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Product_Revenue
    SET Total_Revenue = Total_Revenue + NEW.Amount
    WHERE SKU = NEW.Product_SKU;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Update_Product_Revenue
AFTER INSERT ON Purchased
FOR EACH ROW
EXECUTE FUNCTION Update_Product_Revenue();
```

Code Snippet 11.83 : *Update Product Revenue Trigger*

This code defines a PostgreSQL trigger and its associated function to automatically update the total revenue for a product whenever a new order is placed. The Update_Product_Revenue function increments the Total_Revenue in the Product_Revenue table by the amount of the newly inserted order (NEW.Amount) for the corresponding product (NEW.Product_SKU). This trigger is set to fire after an insert operation on the Purchased table, ensuring that the revenue data remains accurate and up-to-date without manual intervention. This trigger is important because it maintains data integrity and consis-

tency by automating the update process, reducing the risk of human error and ensuring that revenue calculations are always current.

11.3.2 Update Customer Orders Trigger

```
CREATE OR REPLACE FUNCTION Update_Customer_Orders()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Customer_Orders
    SET Total_Orders = Total_Orders + 1
    WHERE Phone_number = NEW.Customer_phone_number;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Update_Customer_Orders
AFTER INSERT ON Orders
FOR EACH ROW
EXECUTE FUNCTION Update_Customer_Orders();
```

Code Snippet 11.84 : *Update Customer Orders Trigger*

This code defines a PostgreSQL trigger and a corresponding function to automatically update the total number of orders placed by each customer whenever a new order is inserted into the Order table. The Update_Customer_Orders function increments the Total_Orders field in the Customer_Orders table for the customer associated with the new order, identified by their phone number. The trigger Update_Customer_Orders is set to execute this function after each new row is inserted into the Order table. This automation ensures that the Total_Orders count remains accurate and up-to-date without requiring manual updates, thereby maintaining data integrity and consistency.

11.3.3 Update Department Employees Trigger

```
CREATE OR REPLACE FUNCTION Update_Department_Employees()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Department_Employees
    SET Total_Employees = Total_Employees + 1
    WHERE Department_name = NEW.Department_name;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Update_Department_Employees
AFTER INSERT ON Employee
FOR EACH ROW
EXECUTE FUNCTION Update_Department_Employees();
```

Code Snippet 11.85 : *Update Department Employees Trigger*

This code defines a PostgreSQL trigger function named Update_Department_Employees that updates the total number of employees in a department whenever a new employee is added. The function increments the Total_Employees field in the Department_Employees table by 1 for the department specified in the NEW.Department_name field, which represents the department of the newly added employee. This trigger is important because it ensures that the Total_Employees count in the Department_Employees table is always accurate and up-to-date, reflecting the current number of employees in each department without requiring manual updates or additional queries. This helps maintain data integrity and consistency within the database.

11.3.4 Update Department Employees Remove Trigger

```
CREATE OR REPLACE FUNCTION Update_Department_Employees_Remove()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Department_Employees
    SET Total_Employees = Total_Employees - 1
    WHERE Department_name = OLD.Department_name;
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Update_Department_Employees_Remove
AFTER DELETE ON Employee
FOR EACH ROW
EXECUTE FUNCTION Update_Department_Employees_Remove();
```

Code Snippet 11.86 : *Update Department Employees Remove Trigger*

This code defines a PostgreSQL trigger and a corresponding function to automatically update the total number of employees in a department whenever an employee is removed from the Employee table. The Update_Department_Employees_Remove function decreases the Total_Employees count in the Department_Employees table by 1 for the department from which the employee was deleted. The trigger Update_Department_Employees_Remove is set to fire after a delete operation on the Employee table, ensuring the department's employee count remains accurate. This automation is useful for maintaining data integrity and consistency without requiring manual updates.

11.3.5 Update Product Revenue Sold Trigger

```
CREATE OR REPLACE FUNCTION Update_Product_Revenue_Sold()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Product_Revenue
    SET Total_Revenue = Total_Revenue + NEW.Amount
    WHERE SKU = NEW.Product_SKU;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
10
11 CREATE TRIGGER Update_Product_Revenue_Sold
12 AFTER INSERT ON Purchased
13 FOR EACH ROW
14 EXECUTE FUNCTION Update_Product_Revenue_Sold();
```

Code Snippet 11.87 : *Update Product Revenue Sold Trigger*

This code defines a PostgreSQL trigger and a corresponding function to automatically update the total revenue for a product whenever a new sale is recorded. The Update_Product_Revenue_Sold function increments the Total_Revenue in the Product_Revenue table by the amount of the newly inserted sale (NEW.Amount) for the product identified by NEW.Product_SKU. This trigger is set to fire after an insert operation on the Purchased table, ensuring that every time a new purchase record is added, the total revenue for the corresponding product is updated accordingly. This mechanism is necessary to maintain accurate and up-to-date revenue data without requiring manual updates, thereby reducing the risk of errors and improving data consistency.

11.4 Functions

11.4.1 Get Product Revenue Function

```
1  DROP FUNCTION get_product_revenue(integer);
2
3  CREATE OR REPLACE FUNCTION Get_Product_Revenue(Product_SK VC)
4  RETURNS NUMERIC AS $$
5  DECLARE
6      Total_Revenue NUMERIC;
7  BEGIN
8      SELECT SUM(Amount) INTO Total_Revenue
9      FROM Purchased
10     WHERE Purchased.Product_SKU = Product_SK;
11     RETURN Total_Revenue;
12 END;
13 $$ LANGUAGE plpgsql;
14
15 SELECT * FROM Get_Product_Revenue('1001');
```

Code Snippet 11.88 : *Get Product Revenue Function*

This SQL script defines a function named Get_Product_Revenue using PL/pgSQL, a procedural language for PostgreSQL. The function calculates the total revenue generated by a specific product identified by its Product_SKU. It does this by summing the Amount from the Purchased table where the Product_SKU matches the input parameter. The result is stored in the Total_Revenue variable and then returned. This function is necessary because it encapsulates the logic for revenue calculation into a reusable and maintainable unit, allowing for consistent and efficient retrieval of revenue data for any product across the database.

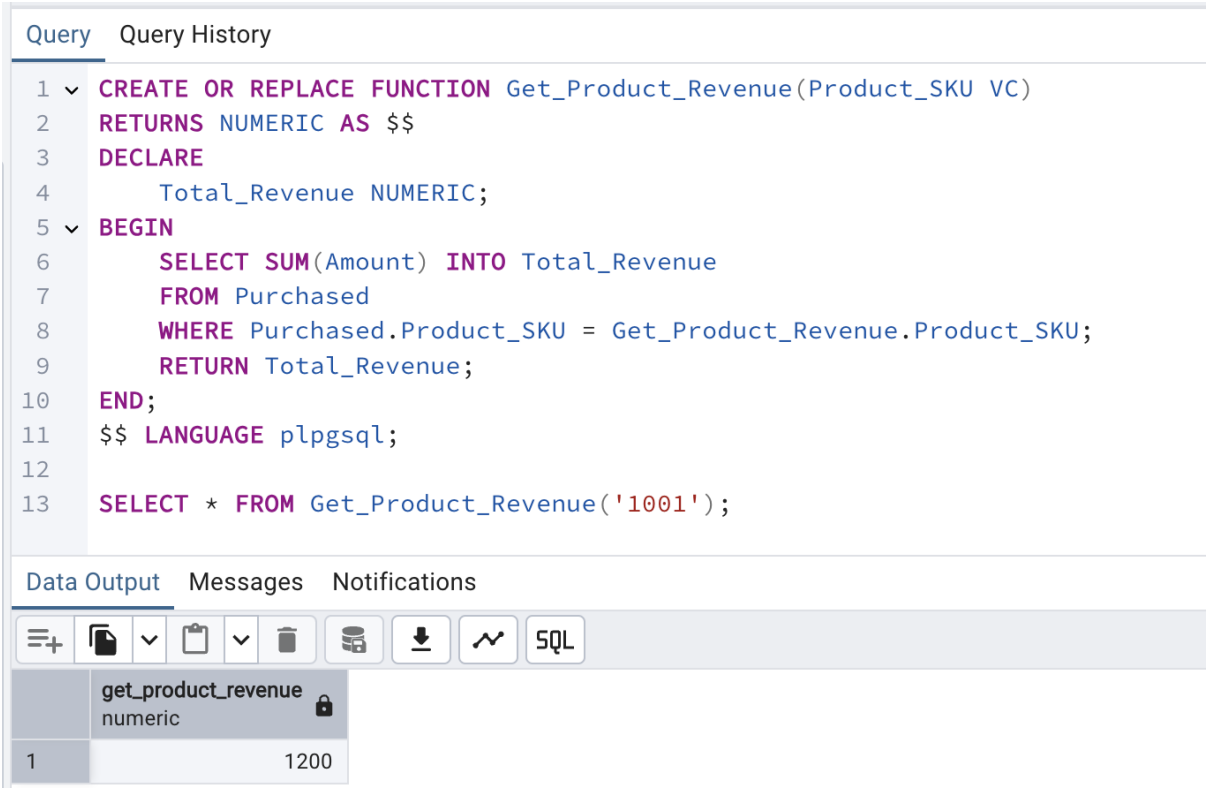


Figure 11.34: *Get Product Revenue Function*

11.4.2 Get Customer Orders Function

```
CREATE OR REPLACE FUNCTION Get_Customer_Orders(Customer_phone_number
    VARCHAR)
RETURNS INT AS $$
DECLARE
    Total_Orders INT;
BEGIN
    SELECT COUNT(Order_id) INTO Total_Orders
    FROM Orders
    WHERE Orders.Customer_phone_number = Get_Customer_Orders.
        Customer_phone_number;
    RETURN Total_Orders;
END;
$$ LANGUAGE plpgsql;

SELECT Get_Customer_Orders('+96134567890');
```

Code Snippet 11.89 : *Get Customer Orders Function*

This SQL code defines a function named Get_Customer_Orders that calculates the total number of orders placed by a specific customer, identified by their phone number. The function takes a customer's phone number as an input parameter and returns an integer representing the total count of orders. Inside the function, a variable Total_Orders is declared to store the result of a SELECT COUNT(Order_id) query, which counts the number of orders associated with the given phone number in the Order table.

This function is useful for quickly retrieving the number of orders for a customer, which can be helpful for customer relationship management, analytics, and reporting purposes. Note that the function uses PL/pgSQL, a procedural language for PostgreSQL.

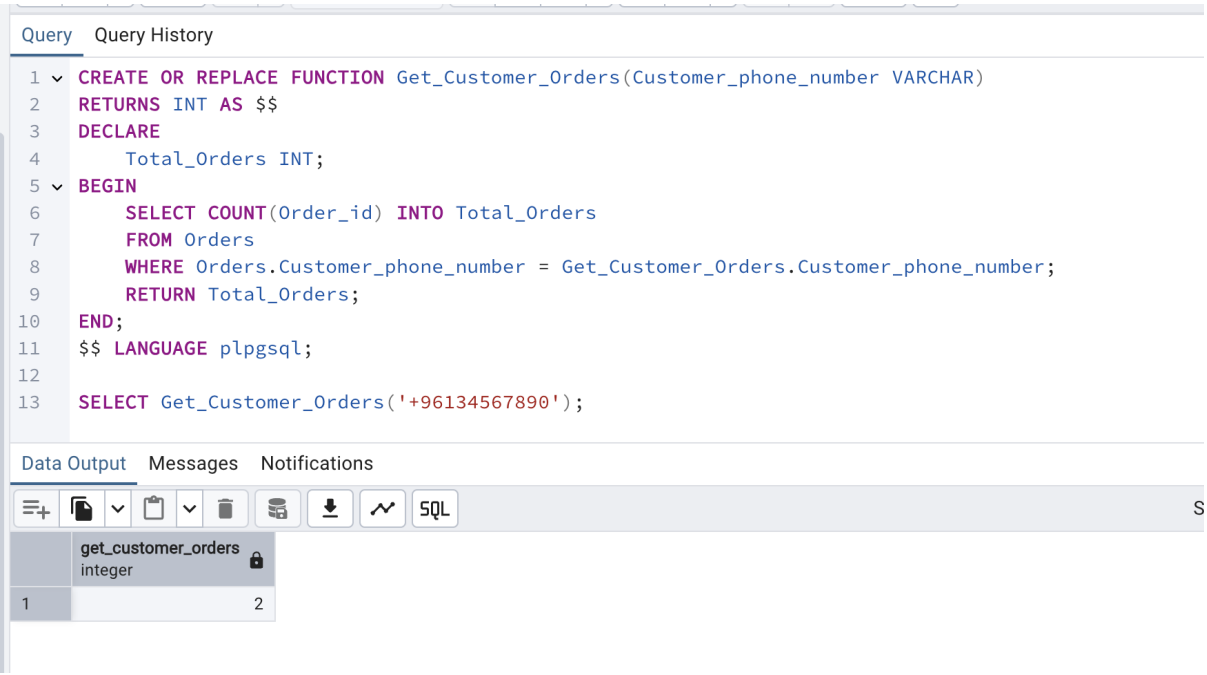


Figure 11.35: *Get Customer Orders Function*

11.4.3 Get Department Employees Function

```
CREATE OR REPLACE FUNCTION Get_Department_Employees(Department_name VARCHAR
    )
RETURNS INT AS $$
DECLARE
    Total_Employees INT;
BEGIN
    SELECT COUNT(SSN) INTO Total_Employees
    FROM Employee
    WHERE Employee.Department_name = Get_Department_Employees.
        Department_name;
    RETURN Total_Employees;
END;
$$ LANGUAGE plpgsql;

SELECT Get_Department_Employees('Sales');
```

Code Snippet 11.90 : *Get Department Employees Function*

This SQL script defines a function named Get_Department_Employees that calculates the total number of employees in a specific department. The function takes a single parameter, Department_name, which is the name of the department for which the employee count is needed. Inside the function, a variable Total_Employees is declared to store the count of employees. The SELECT COUNT(SSN)

INTO Total_Employees statement counts the number of employees (identified by their SSN) in the specified department and stores the result in Total_Employees. Finally, the function returns this count. This function is useful for quickly retrieving the number of employees in any given department, which can be helpful for reporting and analysis purposes in a database management system.

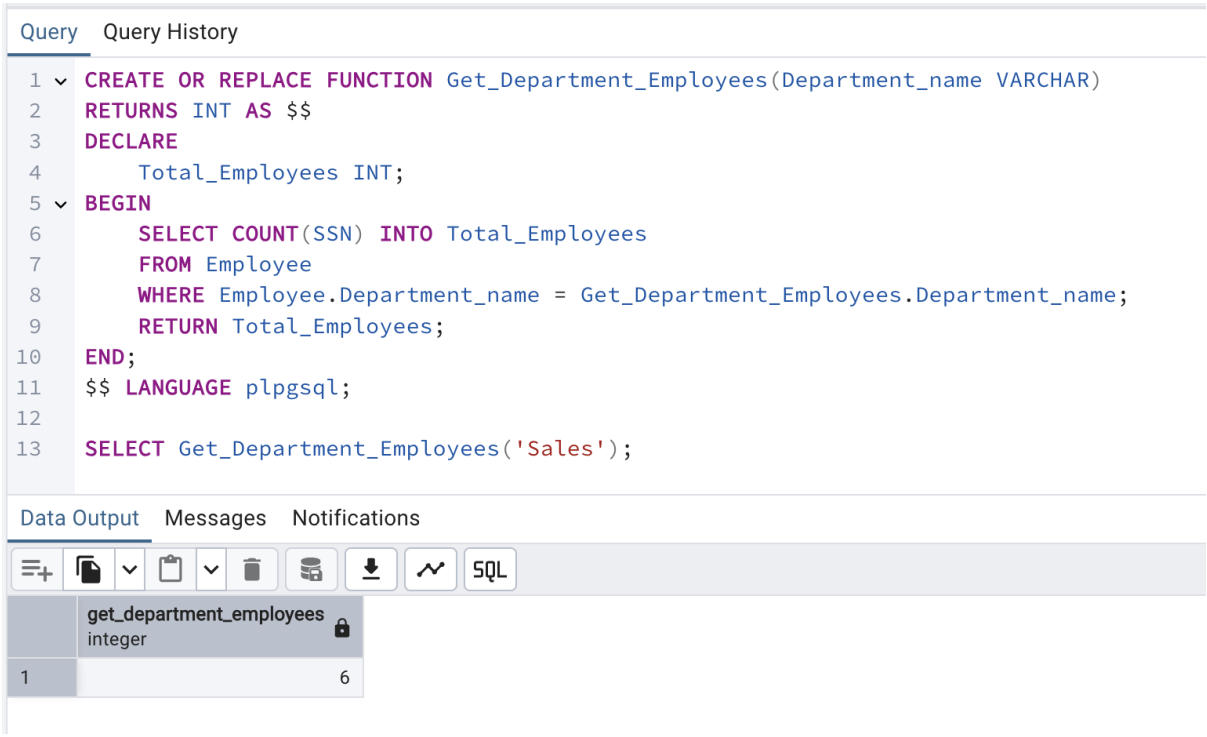


Figure 11.36: *Get Department Employees Function*

11.5 Stored Procedures

11.5.1 Calculate Category Revenue Procedure

```
CREATE OR REPLACE PROCEDURE Calculate_Category_Revenue(Category_name
    VARCHAR)
AS $$
DECLARE
    Total_Revenue NUMERIC;
BEGIN
    SELECT SUM(Purchased.Amount) INTO Total_Revenue
    FROM Purchased
    JOIN Product ON Purchased.Product_SKU = Product.SKU
    WHERE Product.Category_name = Calculate_Category_Revenue.Category_name;
    RAISE NOTICE 'Total revenue generated by category %: %', Category_name,
        Total_Revenue;
END;
$$ LANGUAGE plpgsql;

CALL Calculate_Category_Revenue('Electronics');
```

Code Snippet 11.91 : *Calculate Category Revenue Procedure*

This SQL script creates a stored procedure named Calculate_Category_Revenue in PL/pgSQL, which is a procedural language for PostgreSQL. The procedure takes a single parameter, Category_name, which specifies the product category for which the total revenue needs to be calculated. Inside the procedure, a variable Total_Revenue is declared to store the sum of the amounts from the Purchased table. The procedure performs a SELECT query that joins the Purchased table with the Product table on the Product_SKU field and filters the results by the given Category_name. The total revenue is then stored in the Total_Revenue variable. Finally, the procedure raises a notice displaying the total revenue for the specified category. This query is useful for businesses to quickly calculate and report the total revenue generated by different product categories, aiding in financial analysis and decision-making.

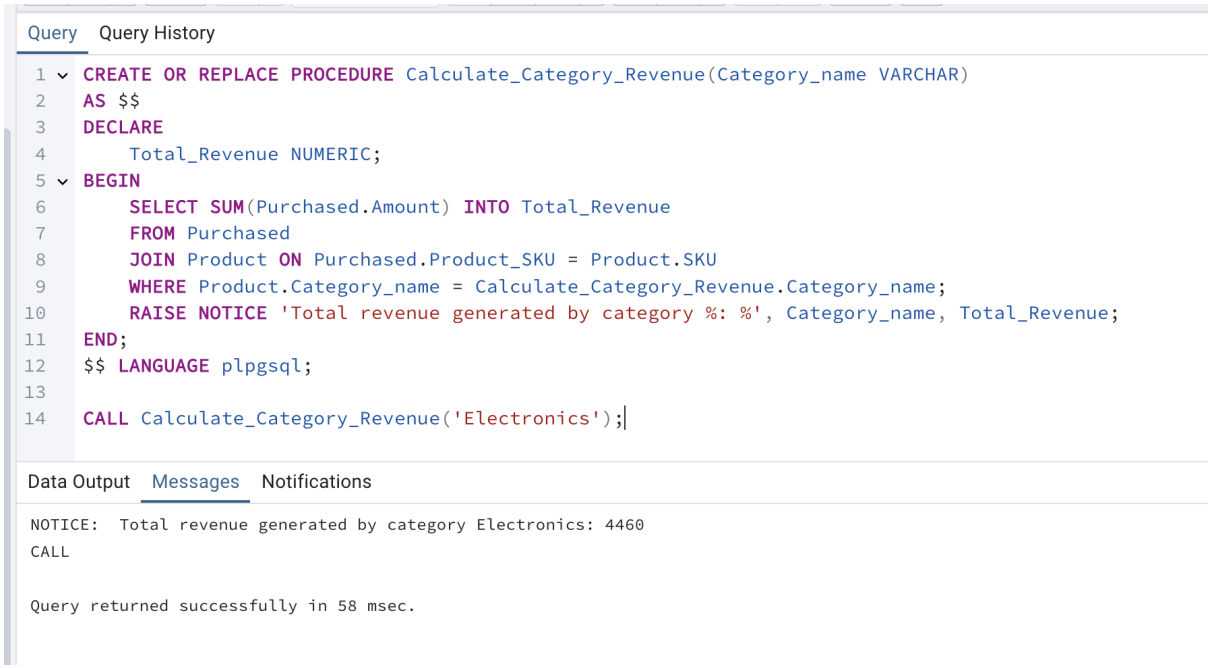


Figure 11.37: *Calculate Category Revenue Procedure*

12 Conclusion

The completion of Phase 3 of the project signifies a major milestone in the learning journey of database systems. By designing and implementing a database in PostgreSQL, we have gained practical experience in utilizing key database management features, including creating and enforcing constraints, writing complex queries, and applying advanced functionalities like triggers and stored procedures. This phase not only consolidates theoretical knowledge but also equips students with the technical skills required to design efficient, secure, and reliable database systems for diverse applications.