



**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 4 of the database project
for the course **EECE433 – Database Systems**

December 2024

Contents

List of Figures	7
List of Tables	9
List of Codes	10
1 Introduction	14
2 References / Copyright Section	14
3 Tool Used to draw the ER Diagram	14
4 System Description & Requirements	14
5 Legend of ER Diagram Symbols	17
6 ER Diagram for the NexStore Database	18
7 New Complete Amended ER Diagram for the NexStore Database	19
7.1 Entity Types and Their Attributes	20
7.1.1 Branch	20
7.1.2 Category	20
7.1.3 Coupon	21
7.1.4 Customer	21
7.1.5 Department	22
7.1.6 Dependent	22
7.1.7 Driver	23
7.1.8 Employee	23
7.1.9 Order	24
7.1.10 Product	25
7.1.11 Supplier	26
7.1.12 Support Ticket	26
7.2 Relationships and their Explanations	27
7.2.1 Assigned To	27
7.2.2 Contains	27
7.2.3 Delivers	28
7.2.4 Dependents Of	28
7.2.5 Is Driver	29
7.2.6 Located In	29
7.2.7 Made By	30
7.2.8 Manages Branch	30
7.2.9 Manages Department	30
7.2.10 Physical Checkout	31
7.2.11 Purchased	31

7.2.12	Redeem	31
7.2.13	Request	32
7.2.14	Reviews	32
7.2.15	Subcategory	33
7.2.16	Supply	33
7.2.17	Supervision	34
7.2.18	Wishlist	34
7.2.19	Works For	35
7.2.20	Works In	35
8	ER to Relational Mapping	35
8.1	Mapping of Strong Entity Types	35
8.1.1	Branch	35
8.1.2	Category	36
8.1.3	Coupon	36
8.1.4	Customer	36
8.1.5	Department	36
8.1.6	Driver	36
8.1.7	Employee	37
8.1.8	Orders	37
8.1.9	Product	37
8.1.10	Supplier	37
8.1.11	Support Ticket	37
8.2	Mapping of Weak Entity Types	38
8.2.1	Dependent	38
8.3	Mapping of Binary 1:1 Relationship Types	38
8.3.1	Redeem	38
8.3.2	Manages Branch	38
8.3.3	Is Driver	39
8.3.4	Manages Department	39
8.4	Mapping of Binary 1:N Relationship Types	39
8.4.1	Subcategory	39
8.4.2	Contains	39
8.4.3	Supply	40
8.4.4	Works In	40
8.4.5	Supervision	40
8.4.6	Physical Checkout	40
8.4.7	Made By	40
8.4.8	Works For	40
8.4.9	Dependents Of	41
8.4.10	Assigned To	41
8.4.11	Delivers	41
8.4.12	Requests	41

8.5	Mapping of Binary M:N Relationship Types	41
8.5.1	Wishlist	42
8.5.2	Located In	42
8.5.3	Reviews	42
8.5.4	Purchased	42
8.6	Mapping of Multivalued Attributes	42
8.6.1	Colors	43
8.6.2	Product Image URLs	43
8.6.3	Review Image URLs	43
8.6.4	Working Hours	43
8.6.5	Department Location	43
9	Final Display – All Tables	44
10	Tables' States	46
11	SQL DDL	63
11.1	Queries	63
11.1.1	Create Queries	63
11.1.1.1	Supplier	63
11.1.1.2	Category	63
11.1.1.3	Product	64
11.1.1.4	Branch	64
11.1.1.5	Department	65
11.1.1.6	Employee	65
11.1.1.7	Driver	66
11.1.1.8	Customer	67
11.1.1.9	Order	68
11.1.1.10	Purchased	68
11.1.1.11	Reviews	69
11.1.1.12	Support Ticket	69
11.1.1.13	Wishlist	70
11.1.1.14	Working hours	70
11.1.1.15	Dependent	71
11.1.1.16	Product Image URLs	71
11.1.1.17	Review Image URLs	72
11.1.1.18	Located in	72
11.1.1.19	Colors	73
11.1.1.20	Coupon	73
11.1.1.21	Department location	74
11.1.2	Alter Queries	74
11.1.2.1	Branch	74
11.1.2.2	Department	74

11.1.3 Insert Queries	74
11.1.3.1 Supplier	74
11.1.3.2 Category	75
11.1.3.3 Product	76
11.1.3.4 Branch & Department & Employee	77
11.1.3.5 Driver	88
11.1.3.6 Customer	88
11.1.3.7 Order	89
11.1.3.8 Purchased	90
11.1.3.9 Reviews	90
11.1.3.10 Support Ticket	91
11.1.3.11 Wishlist	92
11.1.3.12 Working hours	93
11.1.3.13 Dependent	95
11.1.3.14 Product Image URLs	95
11.1.3.15 Review Image URLs	96
11.1.3.16 Located in	96
11.1.3.17 Colors	97
11.1.3.18 Coupon	97
11.1.3.19 Department location	98
11.1.4 Select Queries	99
11.1.4.1 Supplier	99
11.1.4.2 Category	99
11.1.4.3 Product	100
11.1.4.4 Branch	101
11.1.4.5 Department	101
11.1.4.6 Employee	102
11.1.4.7 Driver	103
11.1.4.8 Customer	104
11.1.4.9 Orders	104
11.1.4.10 Purchased	105
11.1.4.11 Reviews	106
11.1.4.12 Support Ticket	107
11.1.4.13 Wishlist	108
11.1.4.14 Working Hours	108
11.1.4.15 Dependent	109
11.1.4.16 Product Image URLs	110
11.1.4.17 Review Image URLs	111
11.1.4.18 Located In	112
11.1.4.19 Colors	113
11.1.4.20 Coupon	114
11.1.4.21 Department Location	115

11.1.5	Complex Queries	116
11.1.5.1	Customer with Highest Number of Orders	116
11.1.5.2	Products with Highest Revenue	118
11.1.5.3	Most Sold Product in Each Branch	119
11.1.5.4	Calculate Customer Lifetime Value for Each Customer	122
11.1.5.5	Underperforming Products	124
11.1.5.6	Find Popular Products Combinations	126
11.1.5.7	Products with the Most Discounts	128
11.1.5.8	Seasonal Trends for Product Categories	129
11.1.5.9	Branches with Stock Shortages	131
11.1.5.10	Best Performing Branch	134
11.2	Views	136
11.2.1	Customer orders	136
11.2.2	Update of the number of employees in the Department	137
11.2.3	Total Revenue Generated by Each Product	139
11.3	Triggers	140
11.3.1	Update Product Revenue Trigger	140
11.3.2	Update Customer Orders Trigger	141
11.3.3	Update Department Employees Trigger	141
11.3.4	Update Department Employees Remove Trigger	142
11.3.5	Update Product Revenue Sold Trigger	142
11.4	Functions	143
11.4.1	Get Product Revenue Function	143
11.4.2	Get Customer Orders Function	144
11.4.3	Get Department Employees Function	145
11.5	Stored Procedures	146
11.5.1	Calculate Category Revenue Procedure	146
12	Normalization	147
12.1	Branch	147
12.2	Category	148
12.3	Colors	148
12.4	Coupon	148
12.5	Customer	149
12.6	Department Table	149
12.7	Department Location Table	150
12.8	Dependent Table	150
12.8.1	D1	150
12.8.2	D2	151
12.9	Driver Table	151
12.10	Employee Table	152
12.11	Located In Table	152
12.12	Orders Table	153

12.13 Product Table	153
12.13.1 P1	154
12.13.2 P2	154
12.13.3 P3	155
12.14 Product Image URLs Table	155
12.15 Purchased Table	156
12.16 Reviews Table	156
12.17 Review Image URLs Table	157
12.18 Support Ticket Table	157
12.19 Supplier Table	158
12.19.1 S1	158
12.19.2 S2	159
12.20 Wishlist Table	159
12.21 Working Hours Table	160
13 Conclusion	160

List of Figures

5.1 <i>Legend of ER Diagram Symbols</i>	17
6.1 <i>ER Diagram for the NexStore Database</i>	18
7.1 <i>ER Diagram for the NexStore Database</i>	19
7.2 <i>Branch Entity and its Attributes</i>	20
7.3 <i>Category Entity and its Attributes</i>	20
7.4 <i>Coupon Entity and its Attributes</i>	21
7.5 <i>Customer Entity and its Attributes</i>	21
7.6 <i>Department Entity and its Attributes</i>	22
7.7 <i>Dependent Entity and its Attributes</i>	22
7.8 <i>Driver Entity and its Attributes</i>	23
7.9 <i>Employee Entity and its Attributes</i>	23
7.10 <i>Order Entity and its Attributes</i>	24
7.11 <i>Product Entity and its Attributes</i>	25
7.12 <i>Supplier Entity and its Attributes</i>	26
7.13 <i>Support Ticket Entity and its Attributes</i>	26
7.14 <i>Assigned To Relationship</i>	27
7.15 <i>Contains Relationship</i>	27
7.16 <i>Delivers Relationship</i>	28
7.17 <i>Dependents Of Relationship</i>	28
7.18 <i>Is Driver Relationship</i>	29
7.19 <i>Located In Relationship</i>	29
7.20 <i>Made By Relationship</i>	30
7.21 <i>Manages Branch Relationship</i>	30
7.22 <i>Manages Department Relationship</i>	30

7.23	<i>Physical Checkout Relationship</i>	31
7.24	<i>Purchased Relationship</i>	31
7.25	<i>Redeem Relationship</i>	31
7.26	<i>Request Relationship</i>	32
7.27	<i>Reviews Relationship</i>	32
7.28	<i>Subcategory Relationship</i>	33
7.29	<i>Supply Relationship</i>	33
7.30	<i>Supervision Relationship</i>	34
7.31	<i>Wishlist Relationship</i>	34
7.32	<i>Works For Relationship</i>	35
7.33	<i>Works In Relationship</i>	35
11.1	<i>Select from Supplier Table</i>	99
11.2	<i>Select from Category Table</i>	100
11.3	<i>Select from Product Table</i>	101
11.4	<i>Select from Branch Table</i>	101
11.5	<i>Select from Department Table</i>	102
11.6	<i>Select from Employee Table</i>	103
11.7	<i>Select from Driver Table</i>	104
11.8	<i>Select from Customer Table</i>	104
11.9	<i>Select from Orders Table</i>	105
11.10	<i>Select from Purchased Table</i>	106
11.11	<i>Select from Reviews Table</i>	107
11.12	<i>Select from Support Ticket Table</i>	107
11.13	<i>Select from Wishlist Table</i>	108
11.14	<i>Select from Working Hours Table</i>	109
11.15	<i>Select from Dependent Table</i>	110
11.16	<i>Select from Product Image URLs Table</i>	111
11.17	<i>Select from Review Image URLs Table</i>	112
11.18	<i>Select from Located In Table</i>	113
11.19	<i>Select from Colors Table</i>	114
11.20	<i>Select from Coupon Table</i>	115
11.21	<i>Select from Department Location Table</i>	116
11.22	<i>Customer with Highest Number of Orders</i>	117
11.23	<i>Products with Highest Revenue</i>	119
11.24	<i>Most Sold Product in Each Branch</i>	121
11.25	<i>Calculate Customer Lifetime Value for Each Customer</i>	123
11.26	<i>Underperforming Products</i>	125
11.27	<i>Find Popular Products Combinations</i>	127
11.28	<i>Products with the Most Discounts</i>	129
11.29	<i>Seasonal Trends for Product Categories</i>	131
11.30	<i>Branches with Stock Shortages</i>	133
11.31	<i>Best Performing Branch</i>	135

11.32	<i>Customer Orders View</i>	137
11.33	<i>Update of the Number of Employees in the Department View</i>	139
11.34	<i>Get Product Revenue Function</i>	144
11.35	<i>Get Customer Orders Function</i>	145
11.36	<i>Get Department Employees Function</i>	146
11.37	<i>Calculate Category Revenue Procedure</i>	147
12.1	<i>Branch Table Normalization</i>	147
12.2	<i>Category Table Normalization</i>	148
12.3	<i>Colors Table Normalization</i>	148
12.4	<i>Coupon Table Normalization</i>	148
12.5	<i>Customer Table Normalization</i>	149
12.6	<i>Department Table Normalization</i>	149
12.7	<i>Department Location Table Normalization</i>	150
12.8	<i>Dependent Table Normalization</i>	150
12.9	<i>D1 Table Normalization</i>	150
12.10	<i>D2 Table Normalization</i>	151
12.11	<i>Driver Table Normalization</i>	151
12.12	<i>Employee Table Normalization</i>	152
12.13	<i>Located In Table Normalization</i>	152
12.14	<i>Orders Table Normalization</i>	153
12.15	<i>Product Table Normalization</i>	153
12.16	<i>P1 Table Normalization</i>	154
12.17	<i>P2 Table Normalization</i>	154
12.18	<i>P3 Table Normalization</i>	155
12.19	<i>Product Image URLs Table Normalization</i>	155
12.20	<i>Purchased Table Normalization</i>	156
12.21	<i>Reviews Table Normalization</i>	156
12.22	<i>Review Image URLs Table Normalization</i>	157
12.23	<i>Support Ticket Table Normalization</i>	157
12.24	<i>Supplier Table Normalization</i>	158
12.25	<i>S1 Table Normalization</i>	158
12.26	<i>S2 Table Normalization</i>	159
12.27	<i>Wishlist Table Normalization</i>	159
12.28	<i>Working Table Normalization</i>	160

List of Tables

9.1	<i>Branch Table</i>	44
9.2	<i>Category Table</i>	44
9.3	<i>Colors Table</i>	44
9.4	<i>Coupon Table</i>	44
9.5	<i>Customer Table</i>	44
9.6	<i>Department Table</i>	44

9.7	<i>Department Location Table</i>	44
9.8	<i>Dependent Table</i>	44
9.9	<i>Driver Table</i>	44
9.10	<i>Employee Table</i>	44
9.11	<i>Located In Table</i>	45
9.12	<i>Orders Table</i>	45
9.13	<i>Product Table</i>	45
9.14	<i>Product Image URLs Table</i>	45
9.15	<i>Purchased Table</i>	45
9.16	<i>Reviews Table</i>	45
9.17	<i>Review Image URLs Table</i>	45
9.18	<i>Support Ticket Table</i>	45
9.19	<i>Supplier Table</i>	45
9.20	<i>Wishlist Table</i>	45
9.21	<i>Working Hours Table</i>	46
10.1	<i>Branch</i>	46
10.2	<i>Category</i>	47
10.3	<i>Colors</i>	48
10.4	<i>Coupon</i>	48
10.5	<i>Customer</i>	49
10.6	<i>Department</i>	49
10.7	<i>Department location</i>	50
10.8	<i>Dependent</i>	51
10.9	<i>Driver</i>	52
10.10	<i>Employee</i>	53
10.11	<i>Located in</i>	54
10.12	<i>Orders</i>	55
10.13	<i>Product</i>	55
10.14	<i>Product Image URLs</i>	56
10.15	<i>Purchased</i>	57
10.16	<i>Reviews</i>	58
10.17	<i>Review Image URLs</i>	59
10.18	<i>Support Ticket</i>	60
10.19	<i>Supplier</i>	60
10.20	<i>Wishlist</i>	61
10.21	<i>Working hours</i>	62

List of Code Snippets

11.1	<i>Create Domains</i>	63
11.2	<i>Create Supplier Table</i>	63
11.3	<i>Create Category Table</i>	63

11.4	<i>Create Product Table</i>	64
11.5	<i>Create Branch Table</i>	64
11.6	<i>Create Department Table</i>	65
11.7	<i>Create Employee Table</i>	65
11.8	<i>Create Driver Table</i>	67
11.9	<i>Create Customer Table</i>	67
11.10	<i>Create Order Table</i>	68
11.11	<i>Create Purchased Table</i>	68
11.12	<i>Create Reviews Table</i>	69
11.13	<i>Create Support Ticket Table</i>	69
11.14	<i>Create Wishlist Table</i>	70
11.15	<i>Create Working Hours Table</i>	70
11.16	<i>DCreate ependent Table</i>	71
11.17	<i>Create Product Image URLs Table</i>	71
11.18	<i>Create Review Image URLs Table</i>	72
11.19	<i>Create Located In Table</i>	72
11.20	<i>Create Colors Table</i>	73
11.21	<i>Create Coupon Table</i>	73
11.22	<i>Create Department Location Table</i>	74
11.23	<i>Alter Branch Table</i>	74
11.24	<i>Alter Department Table</i>	74
11.25	<i>Insert into Supplier Table</i>	74
11.26	<i>Insert into Category Table</i>	75
11.27	<i>Insert into Product Table</i>	76
11.28	<i>Insert into Department Table</i>	77
11.29	<i>Insert into Branch Table</i>	77
11.30	<i>Insert into Employee Table</i>	78
11.31	<i>Update Department Table</i>	81
11.32	<i>Update Branch Table</i>	81
11.33	<i>Insert into Branch, Department, and Employee Tables</i>	82
11.34	<i>Insert into Driver Table</i>	88
11.35	<i>Insert into Customer Table</i>	88
11.36	<i>Insert into Order Table</i>	89
11.37	<i>Insert into Purchased Table</i>	90
11.38	<i>Insert into Reviews Table</i>	90
11.39	<i>Insert into Support Ticket Table</i>	91
11.40	<i>Insert into Wishlist Table</i>	92
11.41	<i>Insert into Working Hours Table</i>	93
11.42	<i>Insert into Dependent Table</i>	95
11.43	<i>Insert into Product Image URLs Table</i>	95
11.44	<i>Insert into Review Image URLs Table</i>	96
11.45	<i>Insert into Located In Table</i>	96

11.46 Insert into Colors Table	97
11.47 Insert into Coupon Table	97
11.48 Insert into Department Location Table	98
11.49 Select from Supplier Table	99
11.50 Select from Category Table	99
11.51 Select from Product Table	100
11.52 Select from Branch Table	101
11.53 Select from Department Table	101
11.54 Select from Employee Table	102
11.55 Select from Driver Table	103
11.56 Select from Customer Table	104
11.57 Select from Orders Table	105
11.58 Select from Purchased Table	105
11.59 Select from Reviews Table	106
11.60 Select from Support Ticket Table	107
11.61 Select from Wishlist Table	108
11.62 Select from Working Hours Table	108
11.63 Select from Dependent Table	109
11.64 Select from Product Image URLs Table	110
11.65 Select from Review Image URLs Table	111
11.66 Select from Located In Table	112
11.67 Select from Colors Table	113
11.68 Select from Coupon Table	114
11.69 Select from Department Location Table	115
11.70 Customer with Highest Number of Orders	116
11.71 Products with Highest Revenue	118
11.72 Most Sold Product in Each Branch	119
11.73 Calculate Customer Lifetime Value for Each Customer	122
11.74 Underperforming Products	124
11.75 Find Popular Products Combinations	126
11.76 Products with the Most Discounts	128
11.77 Seasonal Trends for Product Categories	130
11.78 Branches with Stock Shortages	132
11.79 Best Performing Branch	134
11.80 Customer Orders View	136
11.81 Update of the Number of Employees in the Department View	137
11.82 Total Revenue Generated by Each Product View	139
11.83 Update Product Revenue Trigger	140
11.84 Update Customer Orders Trigger	141
11.85 Update Department Employees Trigger	141
11.86 Update Department Employees Remove Trigger	142
11.87 Update Product Revenue Sold Trigger	142

11.88	<i>Get Product Revenue Function</i>	143
11.89	<i>Get Customer Orders Function</i>	144
11.90	<i>Get Department Employees Function</i>	145
11.91	<i>Calculate Category Revenue Procedure</i>	146

1 Introduction

In this phase of the EECE 433 Database Systems project, the primary objective is to analyze and normalize database schemas to achieve optimal efficiency and integrity. This phase builds upon the foundational work from the earlier phases, focusing on applying normalization techniques to correct and improve the database design. The scope includes identifying functional dependencies, ensuring compliance with various normal forms (1NF, 2NF, 3NF, and BCNF), and restructuring tables to eliminate redundancy and prevent anomalies. Additionally, this phase emphasizes the theoretical and practical application of normalization principles to create well-structured and scalable databases. The project reflects a comprehensive understanding of database normalization and its critical role in designing efficient and reliable database systems, preparing students for real-world database challenges and applications.

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

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for $E_1 : E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

Figure 5.1: Legend of ER Diagram Symbols

6 ER Diagram for the NexStore Database

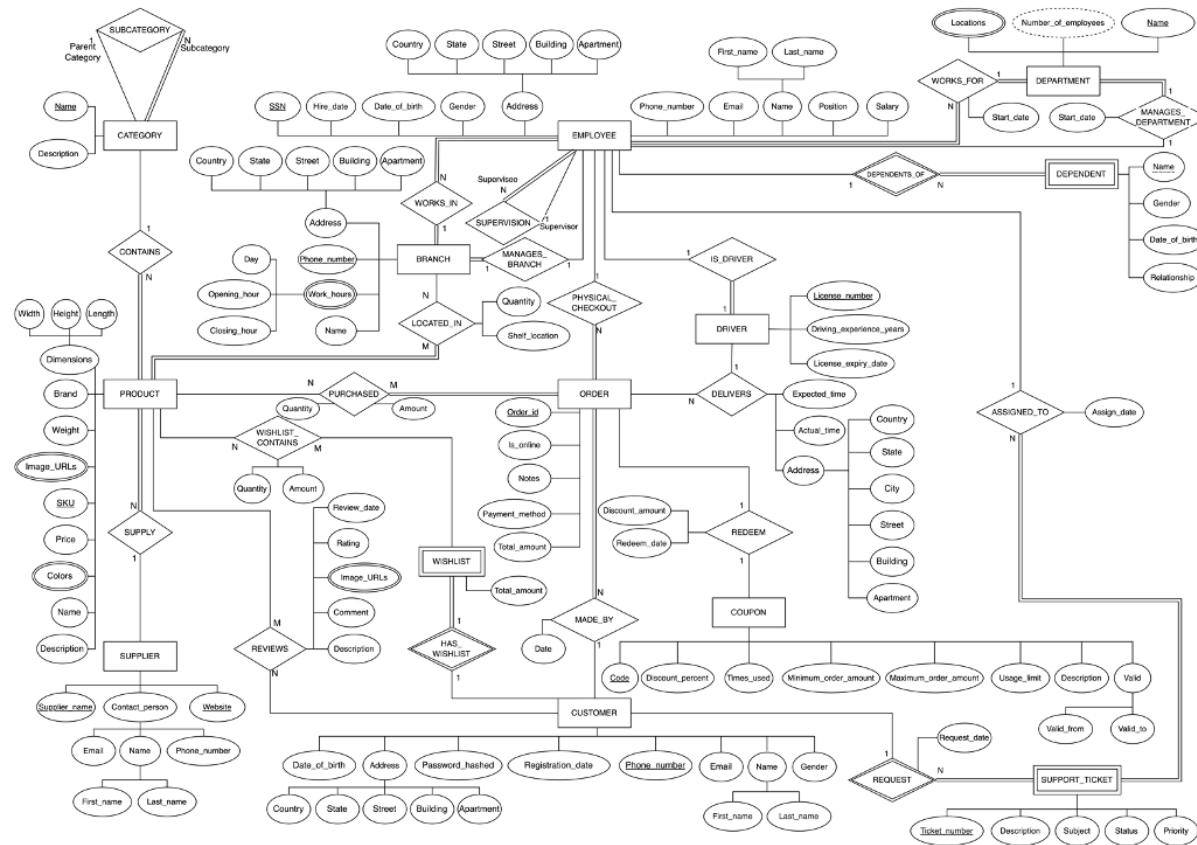


Figure 6.1: ER Diagram for the NexStore Database

7 New Complete Amended ER Diagram for the NexStore Database

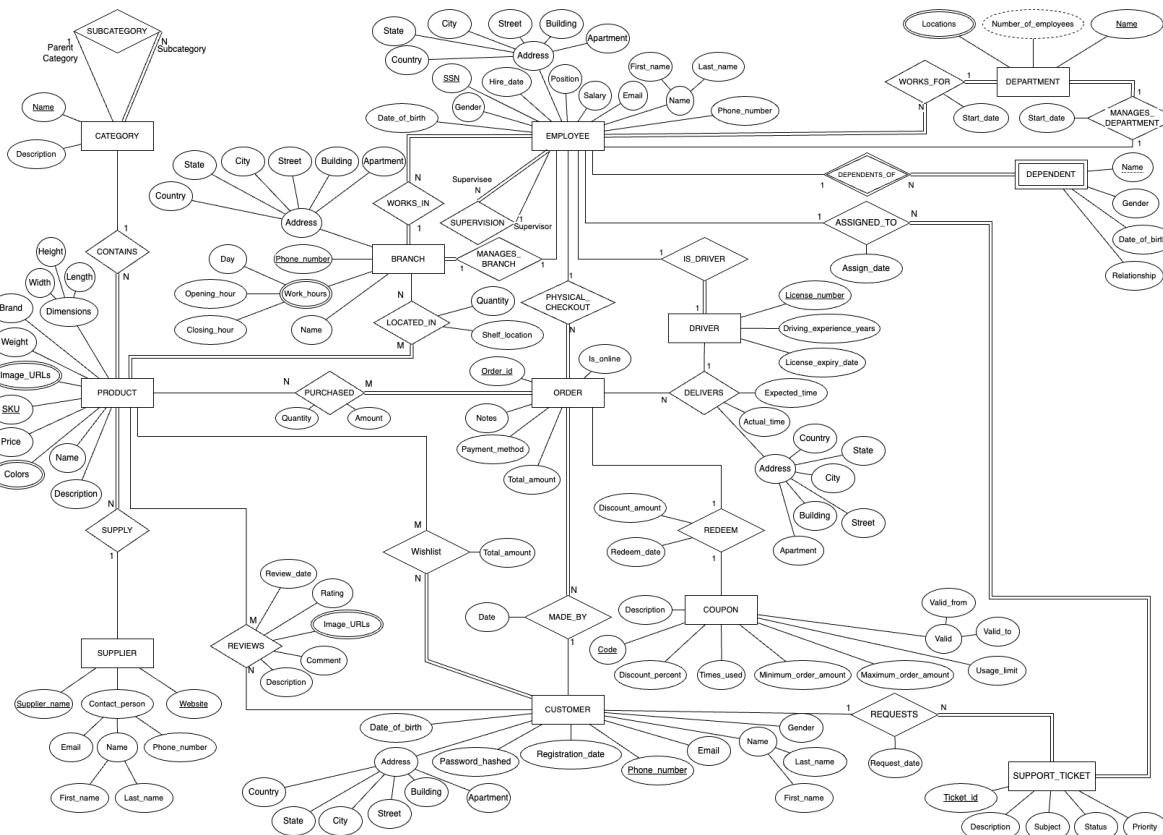


Figure 7.1: ER Diagram for the NexStore Database

7.1 Entity Types and Their Attributes

7.1.1 Branch

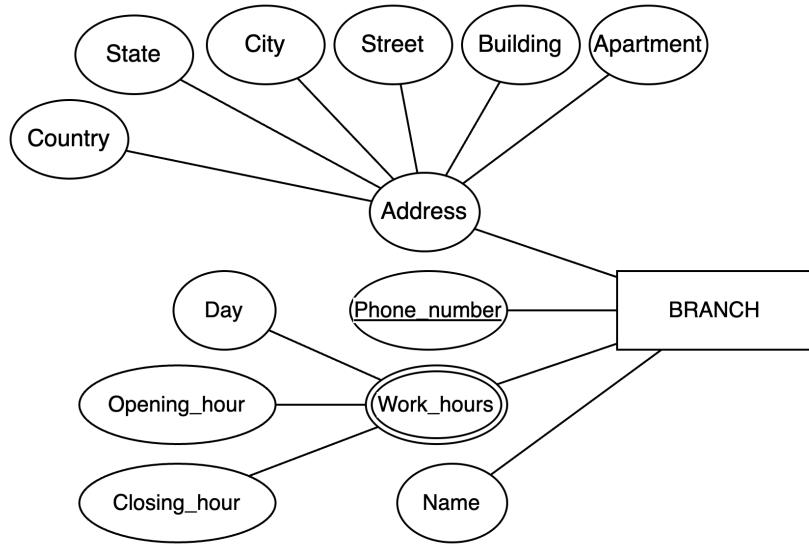


Figure 7.2: *Branch Entity and its Attributes*

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

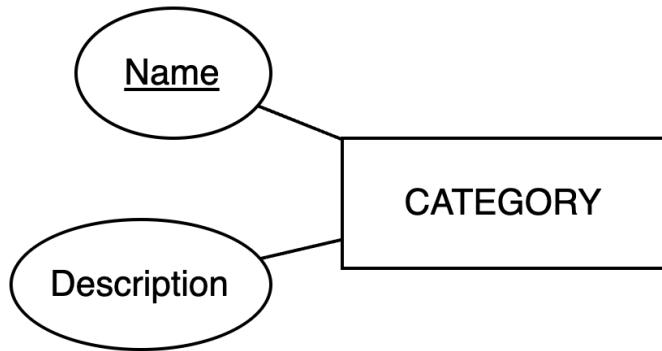


Figure 7.3: *Category Entity and its Attributes*

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

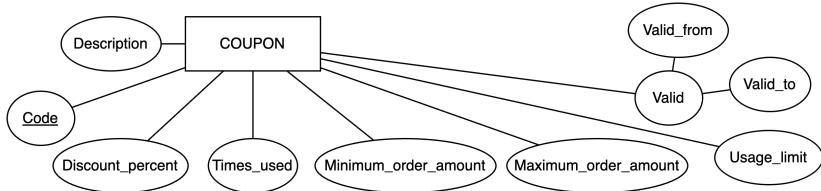


Figure 7.4: *Coupon Entity and its Attributes*

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

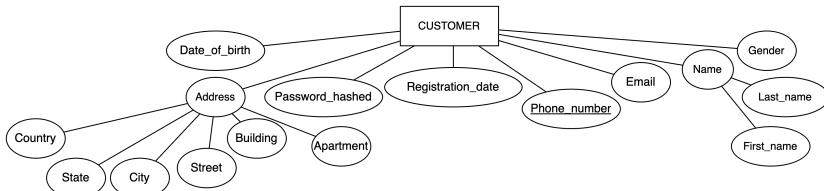


Figure 7.5: *Customer Entity and its Attributes*

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

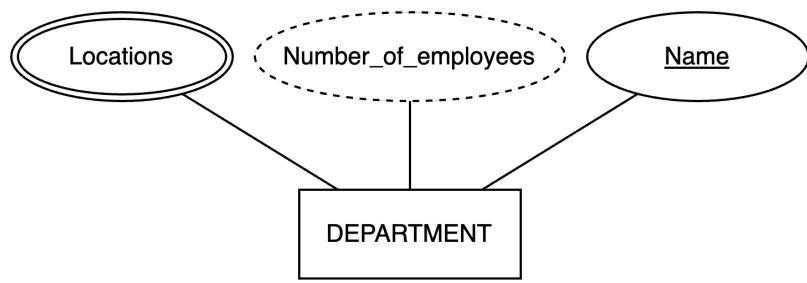


Figure 7.6: *Department Entity and its Attributes*

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

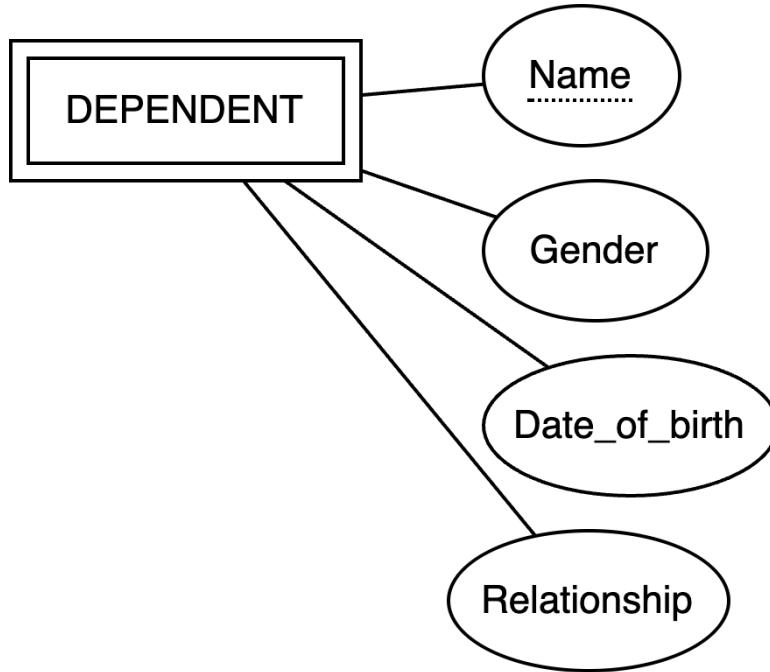


Figure 7.7: *Dependent Entity and its Attributes*

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

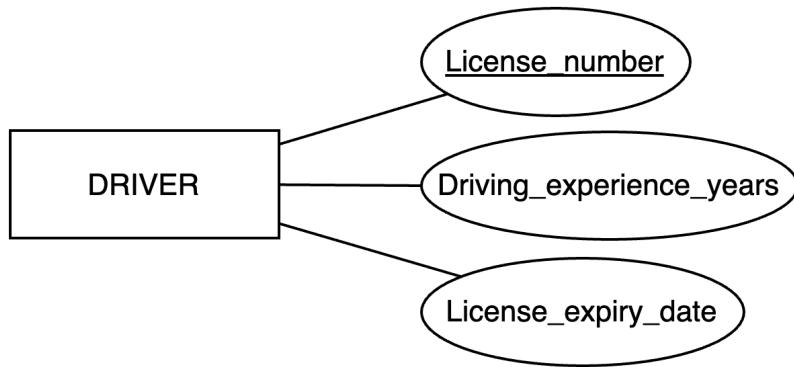


Figure 7.8: *Driver Entity and its Attributes*

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

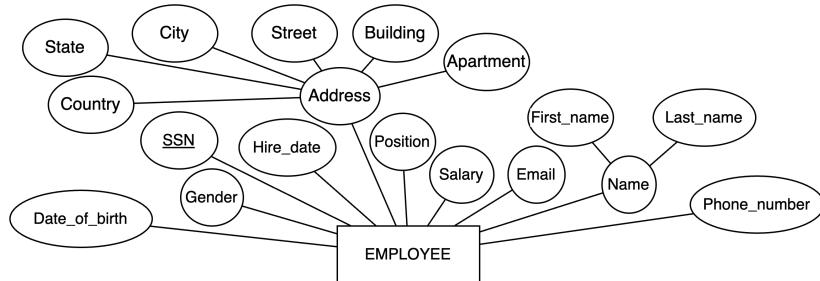


Figure 7.9: *Employee Entity and its Attributes*

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

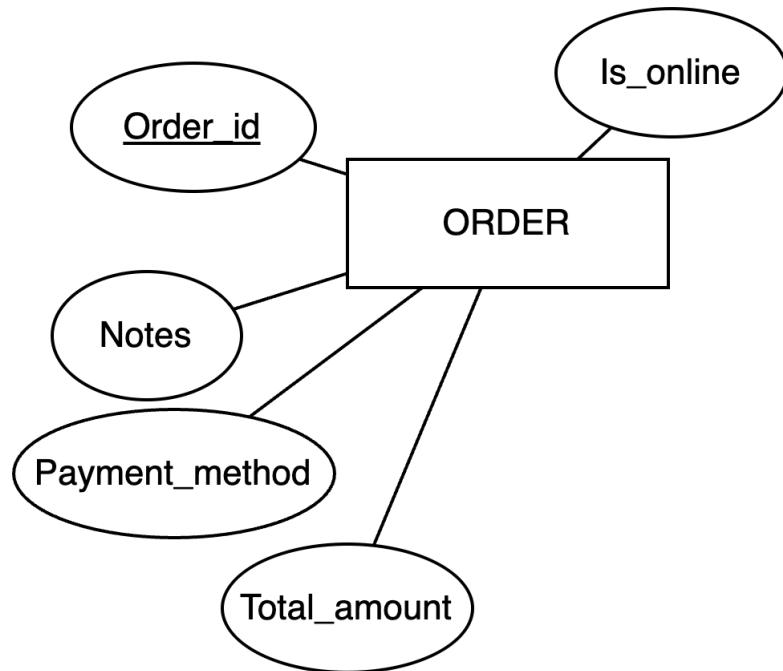


Figure 7.10: *Order Entity and its Attributes*

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

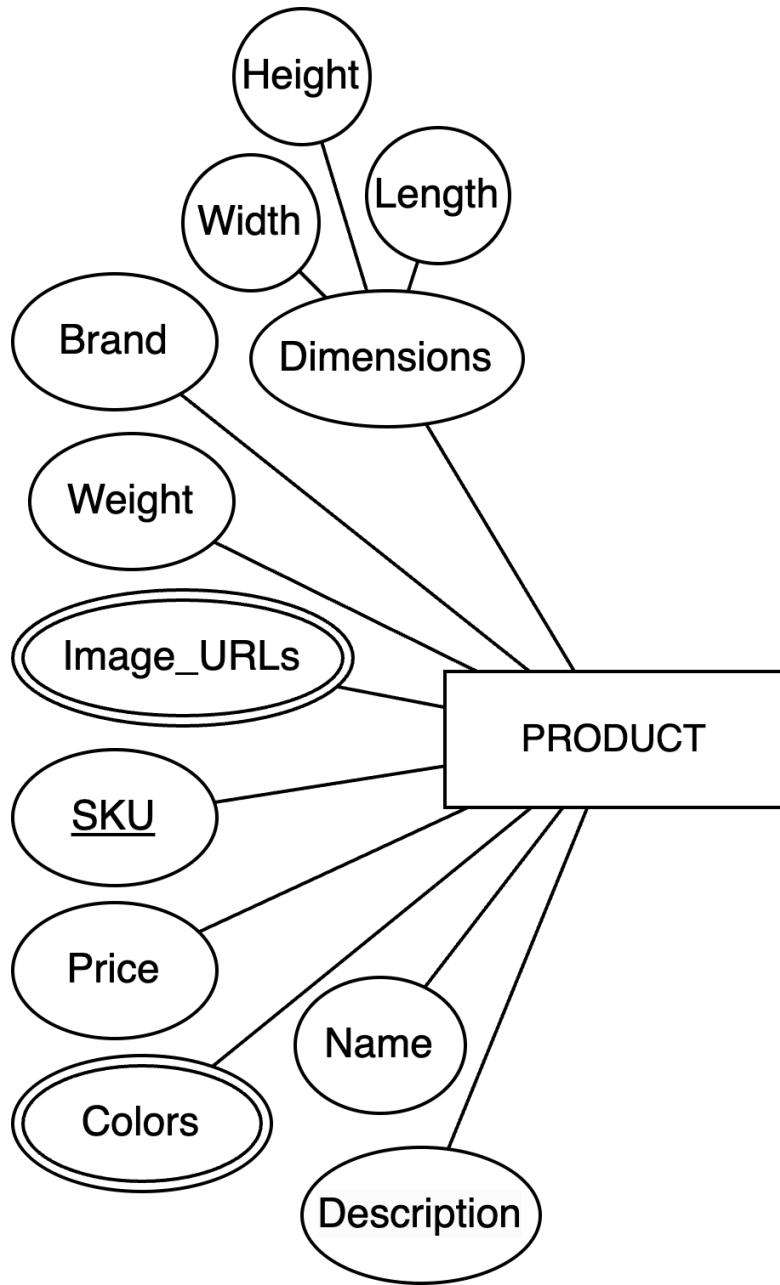


Figure 7.11: *Product Entity and its Attributes*

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

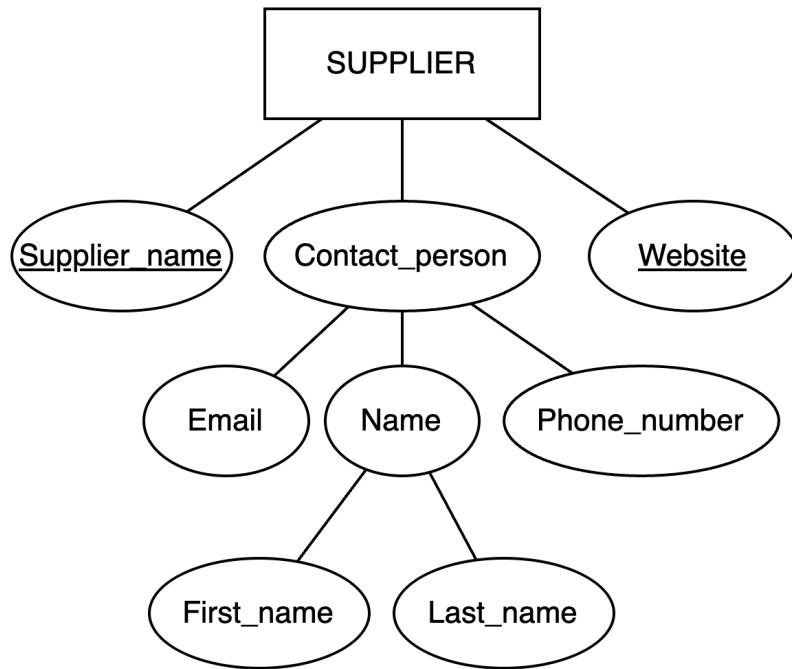


Figure 7.12: *Supplier Entity and its Attributes*

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

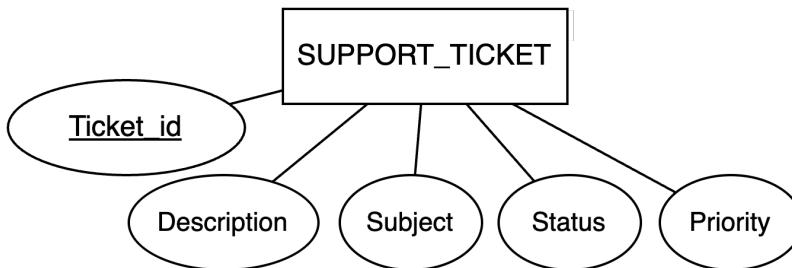


Figure 7.13: *Support Ticket Entity and its Attributes*

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

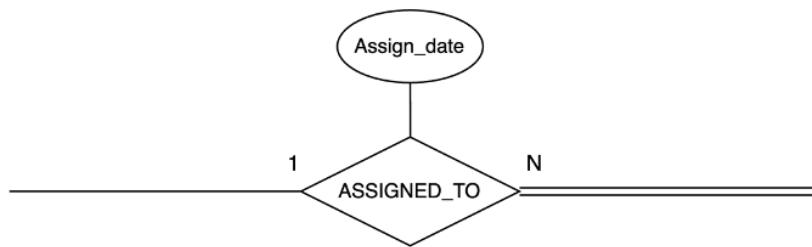


Figure 7.14: *Assigned To Relationship*

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

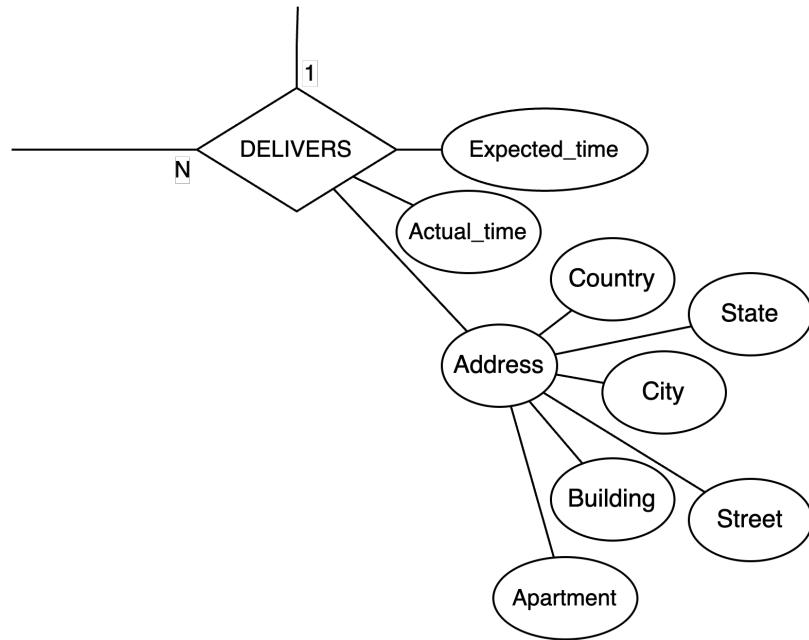


Figure 7.16: *Delivers Relationship*

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

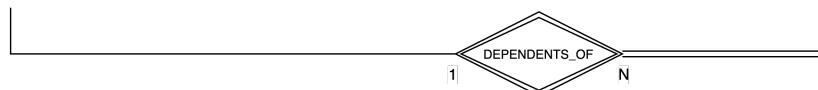


Figure 7.17: *Dependents Of Relationship*

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

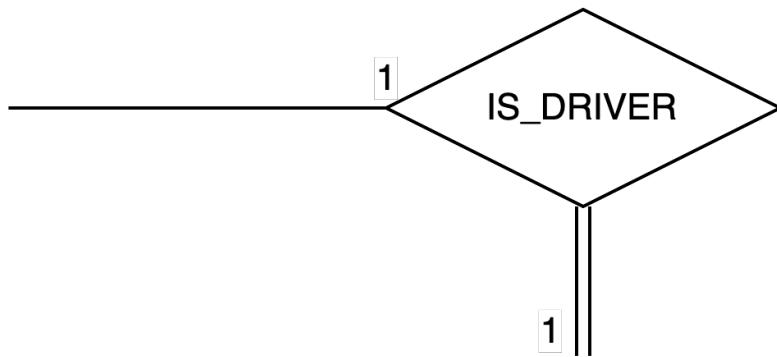


Figure 7.18: *Is Driver Relationship*

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

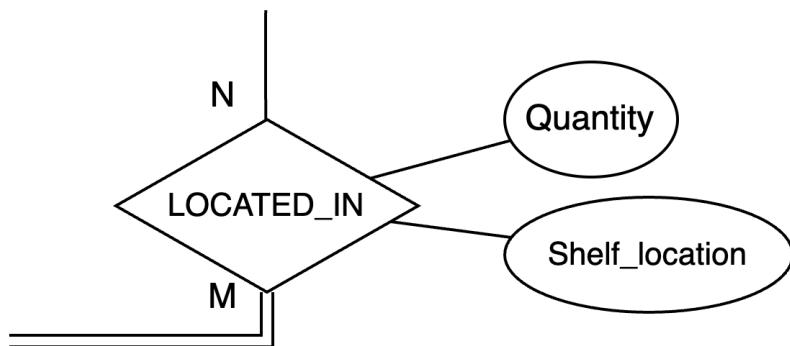


Figure 7.19: *Located In Relationship*

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

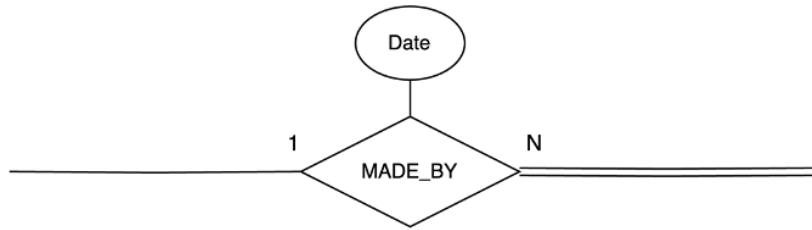


Figure 7.20: *Made By Relationship*

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

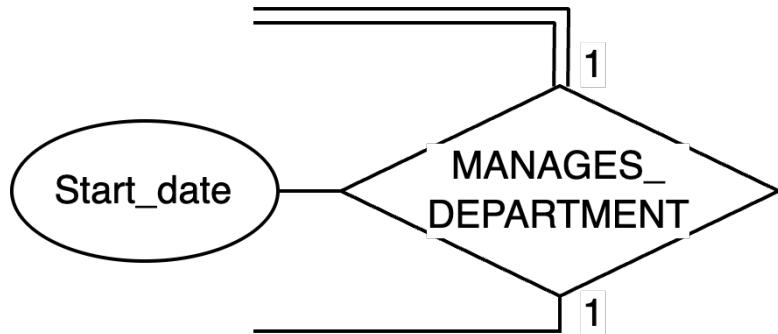


Figure 7.22: *Manages Department Relationship*

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

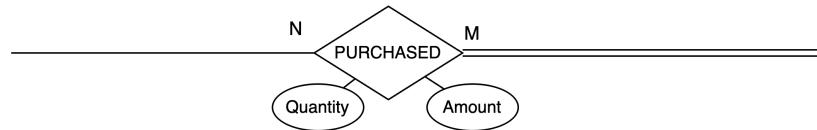


Figure 7.24: *Purchased Relationship*

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

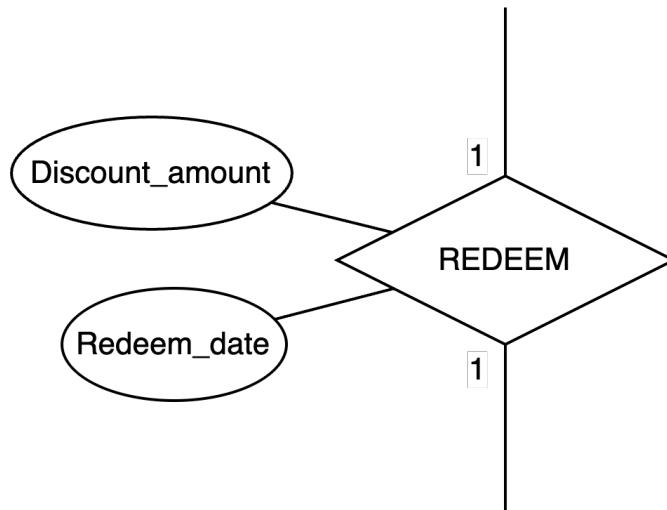


Figure 7.25: *Redeem Relationship*

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

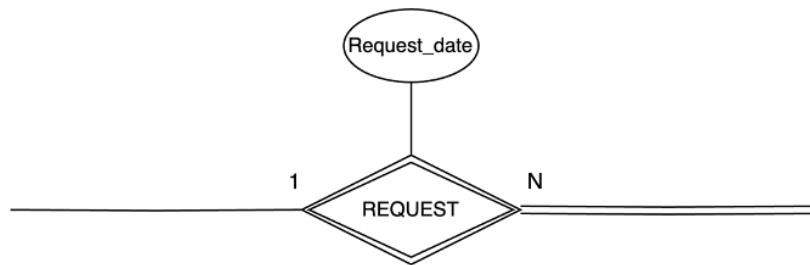


Figure 7.26: Request Relationship

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

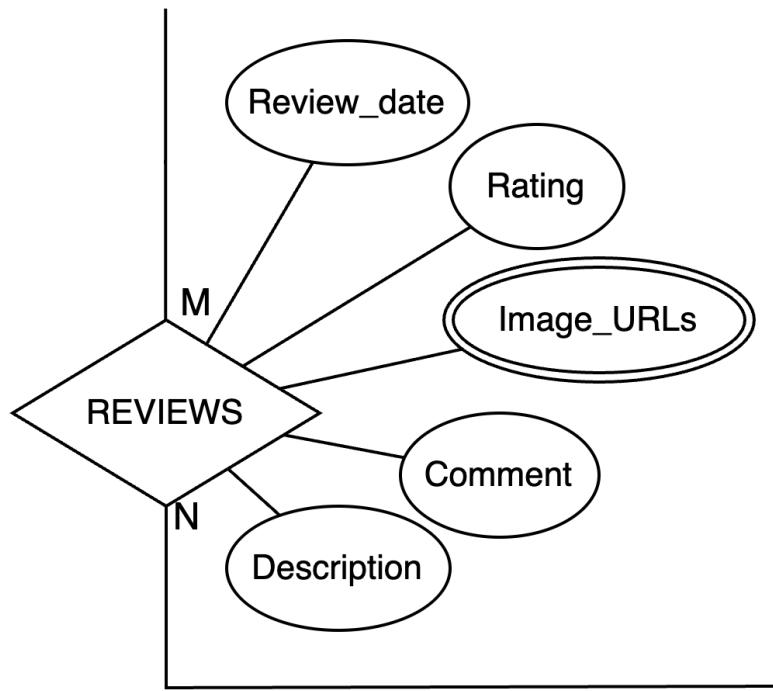


Figure 7.27: Reviews Relationship

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

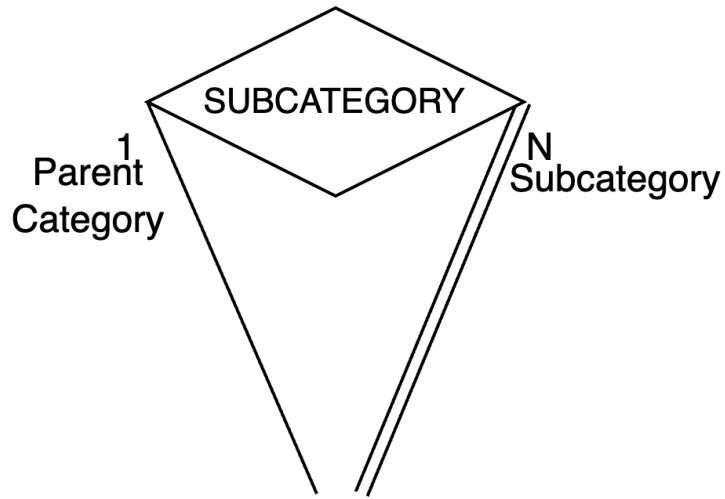


Figure 7.28: *Subcategory Relationship*

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

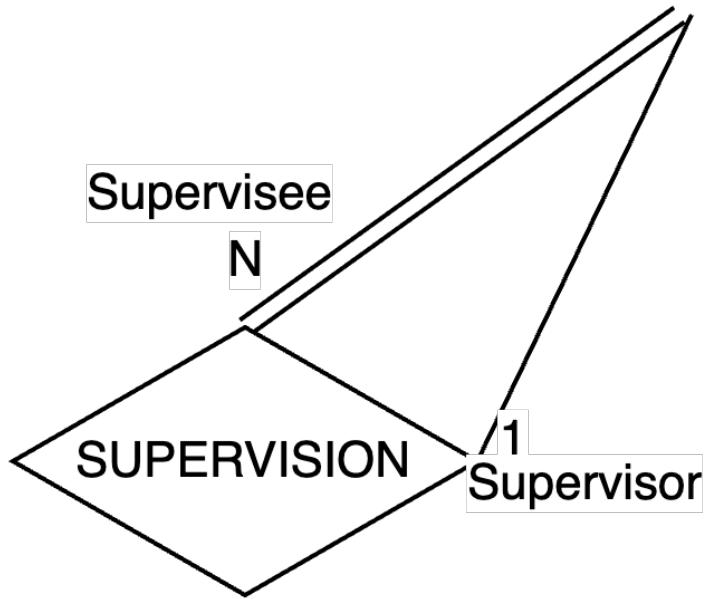


Figure 7.30: *Supervision Relationship*

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

7.2.18 Wishlist

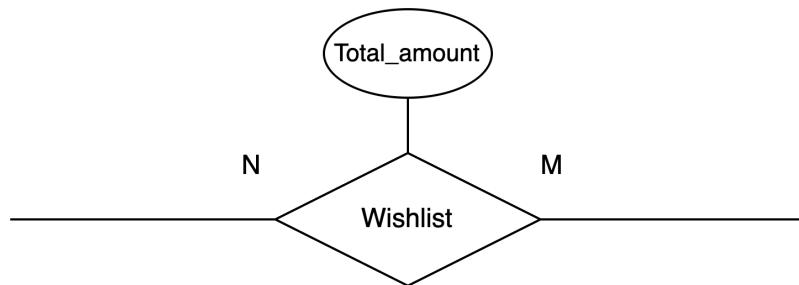


Figure 7.31: *Wishlist Relationship*

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

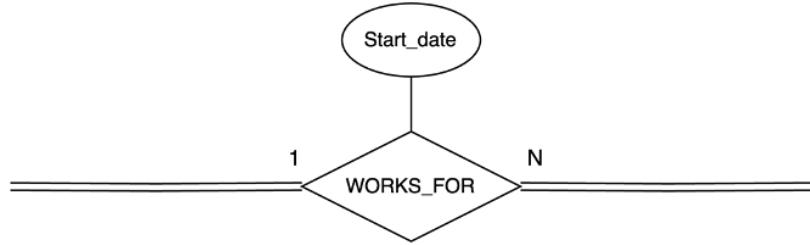


Figure 7.32: *Works For Relationship*

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 Orders

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

We created the relation Orders 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 (Employee_SSN, Name, Gender, Date_of_birth, Relationship)

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 Customer_phone_number to connect the two participating entities.

8.5 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]

8.5.1 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 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]

8.6.1 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 Product Image URLs

Product_Image_URLs (Product_SKU, Product_Image_URL)

The relation Product_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 Product_Image_URLs is the combination of {Product_SKU, Product_Image_URL}.

8.6.3 Review Image URLs

Review_Image_URLs (Product_SKU, Customer_phone_number, Review_Product_Image_URL)

The relation Review_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 attribute Customer_phone_number represents the foreign key of the Customer relation. The primary key of Image_URLs is the combination of {Product_SKU, Review_Product_Image_URL}.

8.6.4 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}.

8.6.5 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*

<u>Phone_number</u>	Name	Country	State	City	Street	Building	Apartment	<i>Employee_SSN</i>
---------------------	------	---------	-------	------	--------	----------	-----------	---------------------

Table 9.2: *Category Table*

<u>Name</u>	Description	<i>Parent_Category_Name</i>
-------------	-------------	-----------------------------

Table 9.3: *Colors Table*

<u>Product_SKU</u>	<u>Product_color</u>
--------------------	----------------------

Table 9.4: *Coupon Table*

<u>Code</u>	Description	Discount_percent	Times_used	Minimum_order_amount	Maximum_order_amount	Usage_limit	Valid_from	Valid_to	<i>Order_ID</i>	Discount_amount	Redeem_date
-------------	-------------	------------------	------------	----------------------	----------------------	-------------	------------	----------	-----------------	-----------------	-------------

Table 9.5: *Customer Table*

<u>Phone_number</u>	Email	First_name	Last_name	Gender	Registration_date	Password_hashed	Date_of_birth	Country	State	City	Street	Building	Apartment
---------------------	-------	------------	-----------	--------	-------------------	-----------------	---------------	---------	-------	------	--------	----------	-----------

Table 9.6: *Department Table*

<u>Name</u>	Number_of_employees	<i>Employee_SSN</i>	Manager_start_date
-------------	---------------------	---------------------	--------------------

Table 9.7: *Department Location Table*

<u>Department_name</u>	<u>Location</u>
------------------------	-----------------

Table 9.8: *Dependent Table*

<u>Employee_SSN</u>	Name	Gender	Date_of_birth	Relationship
---------------------	------	--------	---------------	--------------

Table 9.9: *Driver Table*

<u>License_number</u>	Driving_experience_years	License_expiry_date	<i>Employee_SSN</i>
-----------------------	--------------------------	---------------------	---------------------

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	<i>Branch_phone_number</i>	<i>Supervisor_SSN</i>	<i>Department_name</i>
-----	----------	--------	-----------	--------	---------------	-------	------------	-----------	--------------	---------	-------	------	--------	----------	-----------	----------------------------	-----------------------	------------------------

Table 9.11: *Located In Table*

<u>Product_SKU</u>	<u>Branch_phone_number</u>	Quantity	Shelf_location
--------------------	----------------------------	----------	----------------

Table 9.12: *Orders Table*

<u>Order_id</u>	Notes	Payment_method	Total_amount	Is_online	<u>Employee_SSN</u>	<u>Customer_phone_number</u>	Date	<u>Driver_license_number</u>
-----------------	-------	----------------	--------------	-----------	---------------------	------------------------------	------	------------------------------

Table 9.13: *Product Table*

<u>SKU</u>	Name	Price	Description	Weight	Brand	Width	Height	Length	<u>Category_name</u>	<u>Supplier_website</u>
------------	------	-------	-------------	--------	-------	-------	--------	--------	----------------------	-------------------------

Table 9.14: *Product Image URLs Table*

<u>Product_SKU</u>	<u>Review_Product_Image_URL</u>
--------------------	---------------------------------

Table 9.15: *Purchased Table*

<u>Product_SKU</u>	<u>Order_id</u>	Quantity	Amount
--------------------	-----------------	----------	--------

Table 9.16: *Reviews Table*

<u>Product_SKU</u>	<u>Customer_phone_number</u>	Review_date	Rating	Comment	Description
--------------------	------------------------------	-------------	--------	---------	-------------

Table 9.17: *Review Image URLs Table*

<u>Product_SKU</u>	<u>Customer_phone_number</u>	<u>Product_Image_URL</u>
--------------------	------------------------------	--------------------------

Table 9.18: *Support Ticket Table*

<u>Ticket_id</u>	Description	Subject	Status	Priority	<u>Employee_SSN</u>	<u>Customer_phone_number</u>
------------------	-------------	---------	--------	----------	---------------------	------------------------------

Table 9.19: *Supplier Table*

<u>Website</u>	Supplier.name	Contact_person_email	Contact_person.first_name	Contact_person.last_name	Contact_person.phone_number
----------------	---------------	----------------------	---------------------------	--------------------------	-----------------------------

Table 9.20: *Wishlist Table*

<u>SKU</u>	<u>Customer_phone_number</u>	Total_amount
------------	------------------------------	--------------

Table 9.21: *Working Hours Table*

<u>Branch_phone_number</u>	<u>Day</u>	<u>Opening_hour</u>	<u>Closing_hour</u>
----------------------------	------------	---------------------	---------------------

10 Tables' States

Table 10.1: *Branch*

<u>Phone_number</u>	Name	Country	State	City	Street	Building	Apartment	<i>Employee_SSN</i>
+96111111111	Beirut Main	Lebanon	Beirut	Beirut	Hamra	10	12	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	Beka St	6	4	123459876
+96155555555	Jounieh Store	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	3	567894321
+96166666666	Byblos Outlet	Lebanon	Mount Lebanon	Byblos	Roman Street	6	7	678912345
+96177777777	Tyre Shop	Lebanon	South	Tyre	Port Road	2	1	876543219
+96188888888	Baalbek Point	Lebanon	Beqaa	Baalbek	Temple Rd	3	2	345678912
+96199999999	Batroun Corner	Lebanon	North	Batroun	Old City	1	1	456789123
+96112345678	Downtown Center	Lebanon	Beirut	Beirut	Downtown	9	11	234567891
+96124681357	Dora Warehouse	Lebanon	Mount Lebanon	Dora	Industrial Zone	3	10	789123456
+96165432198	Aley Branch	Lebanon	Mount Lebanon	Aley	Souk Street	6	5	654321987
+96111223344	Choueifat Station	Lebanon	Mount Lebanon	Choueifat	Railway Rd	7	3	987123456

Table 10.2: *Category*

<u>Name</u>	<u>Description</u>	<u>Parent_Category_Name</u>
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

Table 10.3: *Colors*

<u>Product_SKU</u>	<u>Product_color</u>
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*

<u>Code</u>	<u>Description</u>	<u>Discount_percent</u>	<u>Times_used</u>	<u>Minimum_order_amount</u>	<u>Maximum_order_amount</u>	<u>Usage_limit</u>	<u>Valid_from</u>	<u>Valid_to</u>	<u>Order_ID</u>	<u>Discount_amount</u>	<u>Redeem_date</u>
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

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	2022-06-12	*****	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

Table 10.6: *Department*

Name	Number_of_employees	Employee_SSN	Manager_start_date
Sales	6	123456789	2022-03-01
Marketing	2	987654321	2021-06-15
HR	2	123459876	2023-01-10
Finance	3	567894321	2019-10-05
Operations	2	543216789	2020-08-25
IT	1	678912345	2024-04-18
Customer Support	2	876543219	2022-09-12
Logistics	3	345678912	2020-11-20
Legal	1	456789123	2021-02-14
R&D	1	234567891	2022-05-10
Training	4	789123456	2023-03-27
Facilities	1	987123456	2019-06-01

Table 10.7: *Department location*

<u>Department_name</u>	<u>Location</u>
Sales	Beirut
Marketing	Tripoli
HR	Zahle
Finance	Jounieh
Operations	Sidon
IT	Byblos
Customer Support	Tyre
Logistics	Baalbek
Legal	Batroun
R&D	Achrafieh
Training	Dora
Facilities	Antelias

Table 10.8: *Dependent*

<u>Employee_SSN</u>	Name	Gender	Date_of_birth	Relationship
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

Table 10.9: *Driver*

<i>License_number</i>	<i>Driving_experience_years</i>	<i>License_expiry_date</i>	<i>Employee_SSN</i>
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

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	100000	2019-01-15	Male	1980-05-20	john.doe@example.com	John	Doe	+9610000001	Lebanon	Beirut	Hamra	10	12	+96111111111	NULL	Sales	
111111111	AdminAssistant	50000	2020-02-20	Female	1990-08-15	jane.smith@example.com	Jane	Smith	+9610000002	Lebanon	Beirut	Hamra	10	13	+96111111111	123456789	Sales	
111111112	Trainer	60000	2021-05-10	Male	1985-11-30	peter.brown@example.com	Peter	Brown	+9610000003	Lebanon	Beirut	Hamra	10	14	+96111111111	123456789	Sales	
111111113	ITSpecialist	70000	2021-07-01	Male	1992-03-12	alex.jones@example.com	Alex	Jones	+9610000004	Lebanon	Beir	Beirut	Hamra	10	15	+96111111111	123456789	Sales
111111114	ProcurementOfficer	55000	2022-09-15	Female	1995-06-20	lisa.green@example.com	Lisa	Green	+9610000005	Lebanon	Beir	Beirut	Hamra	10	16	+96111111111	123456789	Sales
654321987	AdminAssistant	52000	2021-09-10	Male	1991-06-14	steven.king@example.com	Steven	King	+9610000028	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	+9610000006	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	+9610000007	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	+9610000008	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	+9610000009	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	+9610000010	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	+9610000011	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	+9610000012	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	+9610000013	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	+9610000014	Lebanon	South	Sidon	Corniche	3	7	+96133333333	543216789	Operations
678912345	ITSpecialist	85000	2020-03-10	Male	1983-12-05	kevin.lee@example.com	Kevin	Lee	+9610000015	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	+9610000016	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	+9610000017	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	+9610000018	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	+9610000019	Lebanon	Beqaa	Baalbek	Temple Rd	3	3	+96188888888	345678912	Logistics
888888882	ProcurementOfficer	55000	2019-07-22	Male	1988-12-18	brian.davis@example.com	Brian	Davis	+9610000020	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	+9610000021	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	+9610000022	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	+9610000023	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	+9610000024	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	+9610000025	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	+9610000026	Lebanon	Mount Lebanon	Choueifat	Railway Rd	7	3	+9611223344	NULL	Facilities
234567891	R&DManager	95000	2020-05-10	Female	1983-02-20	rachel.adams@example.com	Rachel	Adams	+9610000027	Lebanon	Beir	Downtown	9	11	+9612345678	NULL	R&D	
023456789	Driver	35000	2020-05-12	Male	1988-03-15	michael.jordan@nexstore.io	Michael	Jordan	+9610000030	Lebanon	Beir	Main Street	5	2	+96111111111	NULL	Logistics	
087654321	Driver	32000	2021-07-18	Female	1992-07-20	emily.clarkson@nexstore.io	Emily	Clarkson	+9610000031	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	+9610000032	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	+9610000033	Lebanon	South	Coastal Road	6	4	+96133333333	NULL	Logistics	
078912345	Driver	34000	2021-12-14	Male	1989-08-25	kevin.baker@nexstore.io	Kevin	Baker	+9610000034	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	+9610000035	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	+9610000036	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	+9610000037	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	+9610000038	Lebanon	Beir	Central Rd	5	4	+9612345678	NULL	Logistics	
089123456	Driver	42000	2016-03-15	Female	1980-10-10	jessica.brooks@nexstore.io	Jessica	Brooks	+9610000039	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	+9610000040	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	+9610000041	Lebanon	Mount Lebanon	Choueifat	Green Lane	2	3	+9611123344	NULL	Logistics
087123456	Driver	40000	2017-07-18	Male	1981-06-02	alex.carter@nexstore.io	Alex	Carter	+9610000042	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	+9610000043	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	+9610000044	Lebanon	Beir	West Blvd	5	2	+96111111111	NULL	Logistics	

Table 10.11: *Located in*

<i>Product_SKU</i>	<i>Branch_phone_number</i>	Quantity	Shelf_location
1001	+961111111111	50	A1
1002	+961222222222	30	B2
1003	+961333333333	20	C3
1004	+961444444444	15	D4
1005	+961555555555	60	E5
1006	+961666666666	25	F6
1007	+961777777777	10	G7
1008	+961888888888	35	H8
1009	+961999999999	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

Table 10.12: *Orders*

<i>Order_id</i>	Notes	Payment_method	Total_amount	Is_online	<i>Employee_SSN</i>	<i>Customer_phone_number</i>	Date	<i>Driver_license_number</i>
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 Year's package	Apple Pay	750	true	111111113	+96178901234	2024-02-03	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

Table 10.13: *Product*

<i>SKU</i>	Name	Price	Description	Weight	Brand	Width	Height	Length	<i>Category_name</i>	<i>Supplier_website</i>
1001	Smartphone	600	5G-enabled phone	0.2kg	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	0.5kg	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	0.25kg	FashionHub	30cm	80cm	1cm	Clothing	www.fashionhub.com
1010	Watch	500	Smartwatch with GPS	0.2kg	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	0.3kg	AudioMax	18cm	20cm	15cm	Electronics	www.audiomax.com
1014	Soccer Ball	40	FIFA approved	0.45kg	SportWear	20cm	20cm	20cm	Sports	www.sportwear.com
1015	Gaming Console	500	Next-gen console	3kg	GameZone	30cm	10cm	25cm	Electronics	www.gamezone.com

Table 10.14: *Product Image URLs*

<u>Product_SKU</u>	<u>Product_Image_URL</u>
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

Table 10.15: *Purchased*

<i>Product_SKU</i>	<i>Order_id</i>	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*

<i>Product_SKU</i>	<i>Customer_phone_number</i>	<i>Review_date</i>	<i>Rating</i>	<i>Comment</i>	<i>Description</i>
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

Table 10.17: *Review Image URLs*

<u>Product_SKU</u>	<u>Customer_phone_number</u>	<u>Product_Image_URL</u>
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

Table 10.18: *Support Ticket*

<u>Ticket_id</u>	Description	Subject	Status	Priority	<i>Employee_SSN</i>	<i>Customer_phone_number</i>
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

Table 10.19: *Supplier*

<u>Website</u>	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.sportwearlb.com	SportWearLebanon	store@sportwear.com	Noah	Miller	+96101234568

Table 10.20: *Wishlist*

<u>SKU</u>	<u>Customer_phone_number</u>	Total_amount
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

Table 10.21: *Working hours*

<u>Branch_phone_number</u>	<u>Day</u>	<u>Opening_hour</u>	<u>Closing_hour</u>
+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	Page 62 of 160	NULL
+96155555555	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.

```
1 CREATE TABLE  
2     IF NOT EXISTS Supplier (  
3         Website VC NOT NULL,  
4         Supplier_name VC NOT NULL,  
5         Contact_person_email VC NOT NULL,  
6         Contact_person_first_name VC NOT NULL,  
7         Contact_person_last_name VC NOT NULL,  
8         Contact_person_phone_number PHONE_NUMBER NOT NULL,  
9         CONSTRAINT PK_Supplier PRIMARY KEY (Website),  
10        CONSTRAINT Supplier_CK_Contact_person_email CHECK (Contact_person_email  
11            LIKE '%@%.%'),  
12        CONSTRAINT Supplier_UK_Supplier_name UNIQUE (Supplier_name),  
13        CONSTRAINT Supplier_UK_Contact_person_phone_number UNIQUE (  
14            Contact_person_phone_number),  
15        CONSTRAINT Supplier_UK_Contact_person_email UNIQUE (  
16            Contact_person_email)  
17    );
```

Code Snippet 11.2 : *Create Supplier Table*

11.1.1.2 Category

The following script creates the Category table.

```
1 CREATE TABLE  
2     IF NOT EXISTS Category (  
3         Name VC NOT NULL,  
4         Description TEXT NOT NULL,
```

```

5 Parent_Category_Name VC,
6 CONSTRAINT PK_Category PRIMARY KEY (Name),
7 CONSTRAINT Category_FK_Category FOREIGN KEY (Parent_Category_Name)
     REFERENCES Category (Name) ON UPDATE CASCADE ON DELETE SET NULL
8 );

```

Code Snippet 11.3 : *Create Category Table*

11.1.1.3 Product

The following script creates the Product table.

```

1 CREATE TABLE
2 IF NOT EXISTS Product (
3     SKU VC NOT NULL,
4     Name VC NOT NULL,
5     Price FLOAT NOT NULL,
6     Description TEXT,
7     Weight FLOAT,
8     Brand VC,
9     Width FLOAT,
10    Height FLOAT,
11    Length FLOAT,
12    Category_name VC,
13    Supplier_website TEXT,
14    CONSTRAINT PK_PRODUCT PRIMARY KEY (SKU),
15    CONSTRAINT Product_FK_Category_name FOREIGN KEY (Category_name)
         REFERENCES Category (Name) ON UPDATE CASCADE ON DELETE SET NULL,
16    CONSTRAINT Product_FK_Supplier_website FOREIGN KEY (Supplier_website)
         REFERENCES Supplier (Website) ON UPDATE CASCADE ON DELETE SET NULL,
17    CONSTRAINT Product_CK_Price CHECK (Price > 0),
18    CONSTRAINT Product_CK_Weight CHECK (Weight > 0),
19    CONSTRAINT Product_CK_Height CHECK (Height > 0),
20    CONSTRAINT Product_CK_Length CHECK (Length > 0),
21    CONSTRAINT Product_UK_Product UNIQUE (Name, Category_name),
22    CONSTRAINT Product_UK_Supplier UNIQUE (Supplier_website, SKU)
23 );

```

Code Snippet 11.4 : *Create Product Table*

11.1.1.4 Branch

The following script creates the Branch table.

```

1 CREATE TABLE
2 IF NOT EXISTS Branch (
3     Phone_number PHONE_NUMBER NOT NULL,
4     Name VC NOT NULL,
5     Country VC NOT NULL,

```

```

6   State VC NOT NULL,
7   City VC NOT NULL,
8   Street VC NOT NULL,
9   Building INT NOT NULL,
10  Apartment INT NOT NULL,
11  Employee_SSN VC,
12  CONSTRAINT PK_Branch PRIMARY KEY (Phone_number),
13  CONSTRAINT Branch_CK_Apartment CHECK (Apartment >= 0),
14  CONSTRAINT Branch_CK_Building CHECK (Building >= 0),
15  CONSTRAINT Branch_UK_LOCATION UNIQUE (Country, State, City, Street,
16    Building, Apartment),
17  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.

```

1 CREATE TABLE
2 IF NOT EXISTS Department (
3   Name VC NOT NULL,
4   Number_of_employees INT NOT NULL,
5   Employee_SSN VC,
6   Manager_start_date DATE NOT NULL,
7   CONSTRAINT PK_Department PRIMARY KEY (Name),
8   CONSTRAINT Department_Department_CK_Number_of_employees CHECK (
9     Number_of_employees > 0),
10  CONSTRAINT Department_Department_CK_Manager_start_date CHECK (
11    Manager_start_date < CURRENT_DATE),
12  CONSTRAINT Department_Department_UK_Manager UNIQUE (Employee_SSN),
13  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.

```

1 CREATE TABLE
2 IF NOT EXISTS Employee (
3   SSN VC NOT NULL,
4   Position VC NOT NULL,
5   Salary INT NOT NULL,
6   Hire_date DATE NOT NULL,
7   Gender VARCHAR(6) NOT NULL,
8   Date_of_birth DATE NOT NULL,

```

```

9   Email VC NOT NULL,
10  First_name VC NOT NULL,
11  Last_name VC NOT NULL,
12  Phone_number PHONE_NUMBER NOT NULL,
13  Country VC NOT NULL,
14  State VC NOT NULL,
15  City VC NOT NULL,
16  Street VC NOT NULL,
17  Building INT NOT NULL,
18  Apartment INT NOT NULL,
19  Branch_phone_number PHONE_NUMBER,
20  Supervisor_SSN VC,
21  Department_name VC NOT NULL,
22  CONSTRAINT PK_EMPLOYEE PRIMARY KEY (SSN),
23  CONSTRAINT Employee_FK_Branch_phone_number FOREIGN KEY (
    Branch_phone_number) REFERENCES Branch (Phone_number) ON UPDATE
    CASCADE ON DELETE SET NULL,
24  CONSTRAINT Employee_FK_Supervisor_SSN FOREIGN KEY (Supervisor_SSN)
    REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL,
25  CONSTRAINT Employee_FK_Department_name FOREIGN KEY (Department_name)
    REFERENCES Department (Name) ON UPDATE CASCADE ON DELETE CASCADE,
26  CONSTRAINT Employee_CK_Gender CHECK (Gender IN ('Male', 'Female')),
27  CONSTRAINT Employee_CK_Date_of_birth CHECK (Date_of_birth <
    CURRENT_DATE),
28  CONSTRAINT Employee_CK_Salary CHECK (Salary > 0),
29  CONSTRAINT Employee_CK_Apartment CHECK (Apartment >= 0),
30  CONSTRAINT Employee_CK_Building CHECK (Building >= 0),
31  CONSTRAINT Employee_CK_Hire_date CHECK (Hire_date < CURRENT_DATE),
32  CONSTRAINT Employee_CK_Email CHECK (Email LIKE '%@%.%'),
33  CONSTRAINT Employee_CK_Position CHECK (
    Position IN ('Manager', 'MarketingHead', 'HRManager', 'FinanceManager',
        'OperationsHead', 'ITSpecialist', 'SupportManager', 'LogisticsHead',
        'LegalAdvisor', 'R&DManager', 'Trainer', 'ProcurementOfficer',
        'AdminAssistant', 'FacilitiesManager', 'ComplianceOfficer', 'Driver')
),
35  CONSTRAINT Employee_UK_Phone_number UNIQUE (Phone_number),
36  CONSTRAINT Employee_UK_Email UNIQUE (Email),
37  CONSTRAINT Employee_UK_SSN UNIQUE (SSN),
38  CONSTRAINT Employee_UK_LOCATION UNIQUE (Country, State, City, Street,
    Building, Apartment)
39 );
40

```

Code Snippet 11.7 : Create Employee Table

11.1.1.7 Driver

The following script creates the Driver table.

```

1 CREATE TABLE
2   IF NOT EXISTS Driver (
3     License_number VC NOT NULL,
4     Driving_experience_years INT NOT NULL,
5     License_expiry_date DATE NOT NULL,
6     Employee_SSN VC NOT NULL,
7     CONSTRAINT PK_DRIVER PRIMARY KEY (License_number),
8     CONSTRAINT Driver_FK_Employee_SSN FOREIGN KEY (Employee_SSN) REFERENCES
9       Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
10      CONSTRAINT Driver_CK_Driving_experience_years CHECK (
11        Driving_experience_years > 0),
12      CONSTRAINT Driver_CK_License_expiry_date CHECK (License_expiry_date >
13        CURRENT_DATE)
14    );

```

Code Snippet 11.8 : *Create Driver Table*

11.1.1.8 Customer

The following script creates the Customer table.

```

1 CREATE TABLE
2   IF NOT EXISTS Customer (
3     Phone_number PHONE_NUMBER NOT NULL,
4     Email VC NOT NULL,
5     First_name VC NOT NULL,
6     Last_name VC NOT NULL,
7     Gender VC,
8     Registration_date DATE NOT NULL,
9     Password_hashed VC NOT NULL,
10    Date_of_birth DATE NOT NULL,
11    Country VC NOT NULL,
12    State VC NOT NULL,
13    City VC NOT NULL,
14    Street VC NOT NULL,
15    Building INT NOT NULL,
16    Apartment INT NOT NULL,
17    CONSTRAINT PK_Customer PRIMARY KEY (Phone_number),
18    CONSTRAINT Customer_CK_Gender CHECK (Gender IN ('Male', 'Female')),
19    CONSTRAINT Customer_CK_Email CHECK (Email LIKE '%@%.%'),
20    CONSTRAINT Customer_CK_Password CHECK (LENGTH (Password_hashed) >= 8),
21    CONSTRAINT Customer_CK_Date_of_birth CHECK (Date_of_birth <
22      CURRENT_DATE),
23    CONSTRAINT Customer_CK_Registration_date CHECK (Registration_date <
24      CURRENT_DATE),
25    CONSTRAINT Customer_CK_Apartment CHECK (Apartment >= 0),
26    CONSTRAINT Customer_CK_Building CHECK (Building >= 0),
27    CONSTRAINT Customer_UK_Email UNIQUE (Email),

```

```

26   CONSTRAINT Customer_UK_Phone_number UNIQUE (Phone_number),
27   CONSTRAINT Customer_UK_LOCATION UNIQUE (Country, State, City, Street,
28     Building, Apartment)
) ;

```

Code Snippet 11.9 : *Create Customer Table*

11.1.1.9 Order

The following script creates the Order table.

```

1 CREATE TABLE
2   IF NOT EXISTS Orders (
3     Order_id ID NOT NULL,
4     Notes TEXT,
5     Payment_method VC NOT NULL,
6     Total_amount INT NOT NULL,
7     Is_online BOOLEAN NOT NULL DEFAULT FALSE,
8     Employee_SSN VC,
9     Customer_phone_number PHONE_NUMBER NOT NULL,
10    Date DATE NOT NULL,
11    Driver_license_number VC,
12    CONSTRAINT PK_Order PRIMARY KEY (Order_id),
13    CONSTRAINT Order_FK_Employee_SSN FOREIGN KEY (Employee_SSN) REFERENCES
14      Employee (SSN) ON UPDATE CASCADE ON DELETE SET NULL,
15    CONSTRAINT Order_FK_Customer_phone_number FOREIGN KEY (
16      Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
17      CASCADE ON DELETE CASCADE,
18    CONSTRAINT Order_FK_Driver_license_number FOREIGN KEY (
19      Driver_license_number) REFERENCES Driver (License_number) ON UPDATE
20      CASCADE ON DELETE SET NULL,
21    CONSTRAINT Order_CK_Is_online CHECK (Is_online IN ('0', '1')),
22    CONSTRAINT Order_CK_Total_amount CHECK (Total_amount > 0),
23    CONSTRAINT Order_CK_Date CHECK (Date < CURRENT_DATE),
24    CONSTRAINT Order_CK_Payment_method CHECK (Payment_method IN ('Cash', '
25      Credit Card', 'Debit Card', 'PayPal', 'Apple Pay', 'Google Pay'))),
26    CONSTRAINT Order_UK_Driver_license_number UNIQUE (Driver_license_number
27      )
28  ) ;

```

Code Snippet 11.10 : *Create Order Table*

11.1.1.10 Purchased

The following script creates the Purchased table.

```

1 CREATE TABLE
2   IF NOT EXISTS Purchased (
3     Product_SKU VC NOT NULL,

```

```

4 Order_id ID NOT NULL,
5 Quantity INT NOT NULL,
6 Amount DECIMAL NOT NULL,
7 CONSTRAINT PK_Purchased PRIMARY KEY (Product_SKU, Order_id),
8 CONSTRAINT Purchased_FK_Product_SKU FOREIGN KEY (Product_SKU)
    REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
9 CONSTRAINT Purchased_FK_Order_id FOREIGN KEY (Order_id) REFERENCES
    Orders (Order_id) ON UPDATE CASCADE ON DELETE CASCADE,
10 CONSTRAINT Purchased_CK_Amount CHECK (Amount >= 0),
11 CONSTRAINT Purchased_CK_Quantity CHECK (Quantity >= 0)
12 );

```

Code Snippet 11.11 : *Create Purchased Table*

11.1.11 Reviews

The following script creates the Reviews table.

```

1 CREATE TABLE
2 IF NOT EXISTS Reviews (
3     Product_SKU VC NOT NULL,
4     Customer_phone_number PHONE_NUMBER NOT NULL,
5     Review_date DATE NOT NULL,
6     Rating INT NOT NULL,
7     Comment TEXT,
8     Description TEXT,
9     CONSTRAINT PK_Reviews PRIMARY KEY (Product_SKU, Customer_phone_number),
10    CONSTRAINT Reviews_FK_Product_SKU FOREIGN KEY (Product_SKU) REFERENCES
        Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
11    CONSTRAINT Reviews_FK_Customer_phone_number FOREIGN KEY (
        Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
        CASCADE ON DELETE CASCADE,
12    CONSTRAINT Reviews_CK_Rating CHECK (
        Rating >= 1
        AND Rating <= 5
    ),
13    CONSTRAINT Reviews_CK_Comment CHECK (Review_date <= CURRENT_DATE)
14 );
15
16
17

```

Code Snippet 11.12 : *Create Reviews Table*

11.1.12 Support Ticket

The following script creates the Support Ticket table.

```

1 CREATE TABLE
2 IF NOT EXISTS Support_Ticket (
3     Ticket_id ID NOT NULL,
4     Description TEXT NOT NULL,

```

```

5   Subject TEXT NOT NULL,
6   Status VARCHAR(8) NOT NULL,
7   Priority VARCHAR(6) NOT NULL,
8   Employee_SSN VC NOT NULL,
9   Customer_phone_number VC NOT NULL,
10  CONSTRAINT PK_Support_Ticket PRIMARY KEY (Ticket_id),
11  CONSTRAINT Support_Ticket_CK_STATUS CHECK (Status IN ('Open', 'Closed',
12    'Resolved')),
12  CONSTRAINT Support_Ticket_CK_PRIORITY CHECK (Priority IN ('High', 'Medium',
13    'Low')),
13  CONSTRAINT Support_Ticket_FK_Employee_SSN FOREIGN KEY (Employee_SSN)
14    REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
14  CONSTRAINT Support_Ticket_FK_Customer_phone_number FOREIGN KEY (
15    Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
15    CASCADE ON DELETE CASCADE
15 );

```

Code Snippet 11.13 : *Create Support Ticket Table*

11.1.1.13 Wishlist

The following script creates the Wishlist table.

```

1 CREATE TABLE
2 IF NOT EXISTS Wishlist (
3   Product_SKU VC NOT NULL,
4   Customer_phone_number PHONE_NUMBER NOT NULL,
5   Total_amount REAL NOT NULL,
6   CONSTRAINT PK_Wishlist PRIMARY KEY (Product_SKU, Customer_phone_number)
7
8   CONSTRAINT Wishlist_FK_Product_SKU FOREIGN KEY (Product_SKU) REFERENCES
9     Product (SKU) ON UPDATE CASCADE ON DELETE SET NULL,
10  CONSTRAINT Wishlist_FK_Customer_phone_number FOREIGN KEY (
11    Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
12    CASCADE ON DELETE CASCADE,
13  CONSTRAINT Wishlist_CK_Total_amount CHECK (Total_amount >= 0)
10 );

```

Code Snippet 11.14 : *Create Wishlist Table*

11.1.1.14 Working hours

The following script creates the Working hours table.

```

1 CREATE TABLE
2 IF NOT EXISTS Working_Hours (
3   Branch_phone_number PHONE_NUMBER NOT NULL,
4   Day VC NOT NULL,
5   Opening_hour TIME,

```

```

6   Closing_hour TIME,
7   CONSTRAINT PK_Working_hours PRIMARY KEY (Branch_phone_number, Day),
8   CONSTRAINT Working_hours_FK_Branch_phone_number FOREIGN KEY (
9     Branch_phone_number) REFERENCES Branch (phone_number) ON UPDATE
10    CASCADE ON DELETE CASCADE,
11   CONSTRAINT Working_hours_CK_Day CHECK (Day IN ('Monday', 'Tuesday', '
12   Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday')),
13   CONSTRAINT Working_hours_CK_Opening_Closing_hour CHECK (
14     (
15       Opening_hour IS NOT NULL
16       AND Closing_hour IS NOT NULL
17     )
18   )
19 )
20 );

```

Code Snippet 11.15 : *Create Working Hours Table*

11.1.1.15 Dependent

The following script creates the Dependent table.

```

1 CREATE TABLE
2 IF NOT EXISTS Dependent (
3   Employee_SSN VC NOT NULL,
4   Name VC NOT NULL,
5   Gender VC NOT NULL,
6   Date_of_birth DATE NOT NULL,
7   Relationship VC NOT NULL,
8   CONSTRAINT PK_DEPENDENT PRIMARY KEY (Employee_SSN, Name),
9   CONSTRAINT Dependent_FK_Employee_SSN FOREIGN KEY (Employee_SSN)
10    REFERENCES Employee (SSN) ON UPDATE CASCADE ON DELETE CASCADE,
11   CONSTRAINT Dependent_CK_Gender CHECK (Gender IN ('Male', 'Female')),
12   CONSTRAINT Dependent_CK_Date_of_birth CHECK (Date_of_birth <
13     CURRENT_DATE)
14 );

```

Code Snippet 11.16 : *DCreate epended Table*

11.1.1.16 Product Image URLs

The following script creates the Product Image URLs table.

```

1 CREATE TABLE
2 IF NOT EXISTS Product_Image_URLs (
3   Product_SKU VC NOT NULL,

```

```

4   Product_Image_URL VC NOT NULL,
5   CONSTRAINT PK_Product_IMAGE_URLS PRIMARY KEY (Product_SKU,
6     Product_Image_URL),
7   CONSTRAINT Product_Image_URLs_FK_Product_Sku FOREIGN KEY (Product_SKU)
8     REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE
9 ;

```

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
1 IF NOT EXISTS Review_Image_URLs (
2   Product_SKU VC NOT NULL,
3   Customer_phone_number PHONE_NUMBER NOT NULL,
4   Product_Image_URL VC NOT NULL,
5   CONSTRAINT PK_Review_IMAGE_URLS PRIMARY KEY (Product_SKU,
6     Product_Image_URL),
7   CONSTRAINT Review_Image_URLs_FK_Product_Sku FOREIGN KEY (Product_SKU)
8     REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
9   CONSTRAINT Review_Image_URLs_FK_Customer_phone_number FOREIGN KEY (
      Customer_phone_number) REFERENCES Customer (Phone_number) ON UPDATE
      CASCADE ON DELETE CASCADE
);

```

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
1 IF NOT EXISTS Located_in (
2   Product_SKU VC NOT NULL,
3   Branch_phone_number PHONE_NUMBER NOT NULL,
4   Quantity INT NOT NULL,
5   Shelf_location VC NOT NULL,
6   CONSTRAINT PK_LOCATED_IN PRIMARY KEY (Product_SKU, Branch_phone_number)
7
8   ,
9   CONSTRAINT Located_in_FK_Product_SKU FOREIGN KEY (Product_SKU)
10    REFERENCES Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE,
11   CONSTRAINT Located_in_FK_Branch_phone_number FOREIGN KEY (
      Branch_phone_number) REFERENCES Branch (Phone_number) ON UPDATE
      CASCADE ON DELETE CASCADE,
12   CONSTRAINT Located_in_CK_Quantity CHECK (Quantity > 0)
);

```

Code Snippet 11.19 : *Create Located In Table*

11.1.19 Colors

The following script creates the Colors table.

```
1 CREATE TABLE
2     IF NOT EXISTS Colors (
3         Product_SKU VC NOT NULL,
4         Product_color VC NOT NULL,
5         CONSTRAINT PK_Colors PRIMARY KEY (Product_SKU, Product_color),
6         CONSTRAINT Colors_FK_Product FOREIGN KEY (Product_SKU) REFERENCES
7             Product (SKU) ON UPDATE CASCADE ON DELETE CASCADE
8     );
```

Code Snippet 11.20 : *Create Colors Table*

11.1.20 Coupon

The following script creates the Coupon table.

```
1 CREATE TABLE
2     IF NOT EXISTS Coupon (
3         Code ID NOT NULL,
4         Description TEXT NOT NULL,
5         Discount_percent REAL NOT NULL,
6         Times_used INT NOT NULL,
7         Minimum_order_amount REAL NOT NULL,
8         Maximum_order_amount REAL NOT NULL,
9         Usage_limit INT NOT NULL,
10        Valid_from DATE NOT NULL,
11        Valid_to DATE NOT NULL,
12        Order_ID ID NOT NULL,
13        Discount_amount REAL NOT NULL,
14        Redeem_date DATE NOT NULL,
15        CONSTRAINT PK_Coupon PRIMARY KEY (Code),
16        CONSTRAINT Coupon_FK_Order FOREIGN KEY (Order_ID) REFERENCES Orders (
17            Order_id) ON UPDATE CASCADE ON DELETE CASCADE,
18        CONSTRAINT Coupon_CK_Percent CHECK (
19            Discount_percent > 0
20            AND Discount_percent <= 100
21        ),
22        CONSTRAINT Coupon_CK_Limit CHECK (Usage_limit > 0),
23        CONSTRAINT Coupon_CK_Valid CHECK (Valid_from <= Valid_to),
24        CONSTRAINT Coupon_CK_Amount CHECK (Discount_amount >= 0),
25        CONSTRAINT Coupon_CK_Times_used CHECK (Times_used >= 0),
26        CONSTRAINT Coupon_CK_Order_amount CHECK (
27            Minimum_order_amount > 0
28            AND Maximum_order_amount > 0
29            AND Minimum_order_amount <= Maximum_order_amount
30        ),
31        CONSTRAINT Coupon_CK_Redeem_date CHECK (Redeem_date <= CURRENT_DATE),
```

```

31   CONSTRAINT Coupon_UK_ORDER UNIQUE (Order_ID)
32 ) ;

```

Code Snippet 11.21 : *Create Coupon Table*

11.1.21 Department location

The following script creates the Department location table.

```

1 CREATE TABLE
2 IF NOT EXISTS Department_Location (
3     Department_name VC NOT NULL,
4     Location VC NOT NULL,
5     CONSTRAINT PK_Department_Location PRIMARY KEY (Department_name,
6             Location),
7     CONSTRAINT Department_Location_FK_Department FOREIGN KEY (
8         Department_name) REFERENCES Department (Name) ON UPDATE CASCADE ON
9         DELETE CASCADE
7 ) ;

```

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.

```

1 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.

```

1 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.

```

1 INSERT INTO
2     Supplier (Website, Supplier_name, Contact_person_email,
3             Contact_person_first_name, Contact_person_last_name,
4             Contact_person_phone_number)

```

```

3 | VALUES
4 | ('www.techbrand.com', 'TechBrand', 'contact@techbrand.com', 'Alice', 'Doe
5 |     ', '+96134567890'),
6 | ('www.computex.com', 'ComputeX', 'support@computex.com', 'John', 'Smith',
7 |     '+96198765432'),
8 | ('www.comfortco.com', 'ComfortCo', 'info@comfortco.com', 'Emma', 'Johnson
9 |     ', '+96145678901'),
10 | ('www.sportwear.com', 'SportWear', 'sales@sportwear.com', 'Michael', 'Brown',
11 |     '+96167890123'),
12 | ('www.musicpro.com', 'MusicPro', 'music@musicpro.com', 'Sarah', 'Lee',
13 |     '+96178901234'),
14 | ('www.homeappl.com', 'HomeAppl', 'appliances@homeappl.com', 'David', 'Evans',
15 |     '+96189012345'),
16 | ('www.visionco.com', 'VisionCo', 'vision@visionco.com', 'Jessica', 'Taylor',
17 |     '+96190123456'),
18 | ('www.kitchenx.com', 'KitchenX', 'service@kitchenx.com', 'Kevin', 'Wilson',
19 |     ', '+96101234567'),
20 | ('www.fashionhub.com', 'FashionHub', 'hello@fashionhub.com', 'Emily', 'Harris',
21 |     '+96123456789'),
22 | ('www.timekeep.com', 'TimeKeep', 'contact@timekeep.com', 'Daniel', 'Martin',
23 |     '+96156789012'),
24 | ('www.edubooks.com', 'EduBooks', 'edu@edubooks.com', 'Olivia', 'White',
25 |     '+96167890124'),
26 | ('www.lightpro.com', 'LightPro', 'light@lightpro.com', 'Robert', 'Scott',
27 |     '+96178901235'),
28 | ('www.audiomax.com', 'AudioMax', 'audio@audiomax.com', 'Sophia', 'Anderson',
29 |     '+96189012346'),
30 | ('www.gamezone.com', 'GameZone', 'games@gamezone.com', 'Liam', 'Thomas',
31 |     '+96190123457'),
32 | ('www.sportwearlb.com', 'SportWearLebanon', 'store@sportwear.com', 'Noah',
33 |     , 'Miller', '+96101234568');

```

Code Snippet 11.25 : Insert into Supplier Table

11.1.3.2 Category

The following script inserts data into the Category table.

```

1 | INSERT INTO
2 |     Category (Name, Description, Parent_Category_Name)
3 | VALUES
4 |     ('Electronics', 'Devices and gadgets', NULL),
5 |     ('Clothing', 'Apparel and fashion items', NULL),
6 |     ('Furniture', 'Home and office furniture', NULL),
7 |     ('Fashion', 'Clothing and accessories', NULL),
8 |     ('Books', 'Printed and digital books', 'Stationery'),
9 |     ('Groceries', 'Food and daily supplies', 'Health'),
10 |    ('Sports', 'Sporting equipment and apparel', 'Clothing'),
11 |    ('Beauty', 'Cosmetics and skincare products', 'Health'),

```

```

12 ('Toys', 'Toys for kids and adults', 'Sports'),
13 ('Automotive', 'Car parts and accessories', 'Electronics'),
14 ('Jewelry', 'Watches, rings, and necklaces', 'Fashion'),
15 ('Health', 'Medical supplies and equipment', 'Beauty'),
16 ('Stationery', 'Office and school supplies', 'Furniture'),
17 ('Pets', 'Pet food and accessories', 'Groceries'),
18 ('Music', 'Instruments and music equipment', 'Art'),
19 ('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.

```

1 INSERT INTO
2   Product (SKU, Name, Price, Description, Weight, Brand, Width, Height,
3             Length, Category_name, Supplier_website)
4 VALUES
5   ('1001', 'Smartphone', 600, '5G-enabled phone', 0.2, 'TechBrand', 7, 15,
6     0.8, 'Electronics', 'www.techbrand.com'),
7   ('1002', 'Laptop', 1200, 'Ultrabook with 16GB RAM', 1.5, 'ComputeX', 32,
8     22, 1.5, 'Electronics', 'www.computex.com'),
9   ('1003', 'Office Chair', 150, 'Ergonomic chair', 12, 'ComfortCo', 60,
10    120, 60, 'Furniture', 'www.comfortco.com'),
11  ('1004', 'Running Shoes', 100, 'Lightweight shoes', 0.5, 'SportWear', 12,
12    35, 10, 'Sports', 'www.sportwear.com'),
13  ('1005', 'Acoustic Guitar', 300, '6-string guitar', 3, 'MusicPro', 38,
14    100, 12, 'Music', 'www.musicpro.com'),
15  ('1006', 'Refrigerator', 800, 'Double door fridge', 65, 'HomeAppl', 90,
16    180, 75, 'Electronics', 'www.homeappl.com'),
17  ('1007', 'LED TV', 400, '50-inch 4K UHD', 8, 'VisionCo', 112, 65, 5, '
18    Electronics', 'www.visionco.com'),
19  ('1008', 'Blender', 70, 'High-speed blender', 2, 'KitchenX', 20, 40, 15,
20    'Electronics', 'www.kitchenx.com'),
21  ('1009', 'T-Shirt', 25, 'Cotton T-shirt', 0.25, 'FashionHub', 30, 80, 1,
22    'Clothing', 'www.fashionhub.com'),
23  ('1010', 'Watch', 500, 'Smartwatch with GPS', 0.2, 'TimeKeep', 5, 5, 1, '
24    Jewelry', 'www.timekeep.com'),
25  ('1011', 'Textbook', 50, 'Advanced mathematics book', 1, 'EduBooks', 21,
26    28, 3, 'Books', 'www.edubooks.com'),
27  ('1012', 'Desk Lamp', 30, 'LED desk lamp', 1.5, 'LightPro', 15, 40, 15, '
28    Furniture', 'www.lightpro.com'),
29  ('1013', 'Wireless Headphones', 150, 'Noise-canceling headphones', 0.3, '
30    AudioMax', 18, 20, 15, 'Electronics', 'www.audiomax.com'),
31  ('1014', 'Soccer Ball', 40, 'FIFA approved', 0.45, 'SportWear', 20, 20,
32    20, 'Sports', 'www.sportwear.com'),
33  ('1015', 'Gaming Console', 500, 'Next-gen console', 3, 'GameZone', 30,
34    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

```
1 INSERT INTO
2   Department (Name, Number_of_employees, Employee_SSN,
3               Manager_start_date)
4 VALUES
5   ('Sales', 6, NULL, '2022-03-01'),
6   ('Marketing', 2, NULL, '2021-06-15'),
7   ('HR', 2, NULL, '2023-01-10'),
8   ('Finance', 3, NULL, '2019-10-05'),
9   ('Operations', 2, NULL, '2020-08-25'),
10  ('IT', 1, NULL, '2024-04-18'),
11  ('Customer Support', 2, NULL, '2022-09-12'),
12  ('Logistics', 3, NULL, '2020-11-20'),
13  ('Legal', 1, NULL, '2021-02-14'),
14  ('Training', 4, NULL, '2023-03-27'),
15  ('Facilities', 1, NULL, '2019-06-01'),
16  ('R&D', 1, NULL, '2022-05-10');
```

Code Snippet 11.28 : Insert into Department Table

Step 2: Insert branches with Employee_SSN as NULL

```
1 INSERT INTO
2   Branch (Phone_number, Name, Country, State, City, Street, Building,
3            Apartment, Employee_SSN)
4 VALUES
5   ('+96111111111', 'Beirut Main', 'Lebanon', 'Beirut', 'Beirut', 'Hamra',
6    10, 12, NULL),
7   ('+96122222222', 'Tripoli Branch', 'Lebanon', 'North', 'Tripoli', 'Mina Street',
8    5, 12, NULL),
9   ('+96133333333', 'Sidon Hub', 'Lebanon', 'South', 'Sidon', 'Corniche',
10  3, 6, NULL),
11  ('+96144444444', 'Zahle Branch', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St',
12  6, 4, NULL),
13  ('+96155555555', 'Jounieh Store', 'Lebanon', 'Mount Lebanon',
14  'Jounieh', 'Coastal Rd', 5, 3, NULL),
15  ('+96166666666', 'Byblos Outlet', 'Lebanon', 'Mount Lebanon',
16  'Byblos', 'Roman Street', 6, 7, NULL),
17  ('+96177777777', 'Tyre Shop', 'Lebanon', 'South', 'Tyre', 'Port Road',
18  2, 1, NULL),
19  ('+96188888888', 'Baalbek Point', 'Lebanon', 'Beqaa', 'Baalbek',
20  'Temple Rd', 3, 2, NULL),
```

```

12 ('+96199999999', 'Batroun Corner', 'Lebanon', 'North', 'Batroun', 'Old City', 1, 1, NULL),
13 ('+96112345678', 'Downtown Center', 'Lebanon', 'Beirut', 'Beirut', 'Downtown', 9, 11, NULL),
14 ('+96124681357', 'Dora Warehouse', 'Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 10, NULL),
15 ('+96165432198', 'Aley Branch', 'Lebanon', 'Mount Lebanon', 'Aley', 'Souk Street', 6, 5, NULL),
16 ('+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

```

1 INSERT INTO
2 Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth,
3 Email, First_name, Last_name, Phone_number, Country, State, City,
4 Street, Building, Apartment, Branch_phone_number, Supervisor_SSN
5 , Department_name)
VALUES
6 -- Sales Department Employees
7 (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'),
8 (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'),
9 (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'),
10 (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'),
11 (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'),
12 (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
13 (987654321, 'MarketingHead', 90000, '2018-06-10', 'Female', '1982-09-25', 'sarah.johnson@example.com', 'Sarah', 'Johnson', 'Marketing')

```

```

13      '+96100000006', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5,
12, '+96122222222', NULL, 'Marketing'),
13 (222222222, 'AdminAssistant', 50000, '2021-08-01', 'Male', '1
1990-12-12', 'michael.taylor@example.com', 'Michael', 'Taylor', '+
96100000007', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5,
13, '+96122222222', 987654321, 'Marketing'),
14 -- HR Department Employees
15 (123459876, 'HRManager', 85000, '2023-01-10', 'Male', '1984-03-15',
16     'david.wilson@example.com', 'David', 'Wilson', '+96100000008', 'Lebanon',
17     'Beqaa', 'Zahle', 'Beka St', 6, 4, '+96144444444', NULL,
18     , 'HR'),
19 (333333333, 'AdminAssistant', 48000, '2023-02-15', 'Female', '1
1991-07-22', 'emily.davis@example.com', 'Emily', 'Davis', '+
96100000009', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St', 6, 5, '+
96144444444', 123459876, 'HR'),
20 -- Finance Department Employees
21 (567894321, 'FinanceManager', 95000, '2019-10-05', 'Female', '1
1978-11-08', 'laura.martin@example.com', 'Laura', 'Martin', '+
96100000010', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
22     ', 5, 3, '+96155555555', NULL, 'Finance'),
23 (444444444, 'AdminAssistant', 47000, '2020-03-12', 'Male', '1
1989-04-18', 'daniel.moore@example.com', 'Daniel', 'Moore', '+
96100000011', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
24     ', 5, 4, '+96155555555', 567894321, 'Finance'),
25 (555555555, 'ComplianceOfficer', 62000, '2021-07-23', 'Female', '1
1993-09-30', 'olivia.thomas@example.com', 'Olivia', 'Thomas', '+
96100000012', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd
26     ', 5, 5, '+96155555555', 567894321, 'Finance'),
27 -- Operations Department Employees
28 (543216789, 'OperationsHead', 90000, '2018-08-15', 'Male', '1
1980-07-22', 'mark.stevens@example.com', 'Mark', 'Stevens', '+
96100000013', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 6, '+
96133333333', NULL, 'Operations'),
29 (666666666, 'AdminAssistant', 50000, '2019-05-20', 'Female', '1
1992-01-15', 'anna.miller@example.com', 'Anna', 'Miller', '+
96100000014', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 7, '+
96133333333', 543216789, 'Operations'),
30 -- IT Department Employee
31 (678912345, 'ITSpecialist', 85000, '2020-03-10', 'Male', '1983-12-05
32     ', 'kevin.lee@example.com', 'Kevin', 'Lee', '+96100000015', 'Lebanon',
33     'Mount Lebanon', 'Byblos', 'Roman Street', 6, 7, '+
96166666666', NULL, 'IT'),
34 -- Customer Support Department Employees
35 (876543219, 'SupportManager', 80000, '2017-11-01', 'Female', '1
1985-04-18', 'laura.kim@example.com', 'Laura', 'Kim', '+
96100000016', 'Lebanon', 'South', 'Tyre', 'Port Road', 2, 1, '+
96177777777', NULL, 'Customer Support'),
36 (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'),
29 -- Logistics Department Employees
30 (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'),
31 (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'),
32 (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'),
33 -- Legal Department Employee
34 (456789123, 'LegalAdvisor', 95000, '2016-09-05', 'Male', '1977-11-11
', 'richard.harris@example.com', 'Richard', 'Harris', '
+96100000021', 'Lebanon', 'North', 'Batraoun', 'Old City', 1, 1,
'+96199999999', NULL, 'Legal'),
35 -- Training Department Employees
36 (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'),
37 (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'),
38 (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'),
39 (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'),
40 -- Facilities Department Employee
41 (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'),
42 -- R&D Department Employee
43 (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)

```
1 UPDATE Department
2 SET
3 Employee_SSN = CASE Name
4     WHEN 'Sales' THEN '123456789'
5     WHEN 'Marketing' THEN '987654321'
6     WHEN 'HR' THEN '123459876'
7     WHEN 'Finance' THEN '567894321'
8     WHEN 'Operations' THEN '543216789'
9     WHEN 'IT' THEN '678912345'
10    WHEN 'Customer Support' THEN '876543219'
11    WHEN 'Logistics' THEN '345678912'
12    WHEN 'Legal' THEN '456789123'
13    WHEN 'Training' THEN '789123456'
14    WHEN 'Facilities' THEN '987123456'
15    WHEN 'R&D' THEN '234567891'
16    ELSE Employee_SSN
17
18 END;
```

Code Snippet 11.31 : *Update Department Table*

Step 5: Update branches to set Employee_SSN

```
1 UPDATE Branch
2 SET
3 Employee_SSN = CASE Phone_number
4     WHEN '+96111111111' THEN '123456789'
5     WHEN '+96122222222' THEN '987654321'
6     WHEN '+96133333333' THEN '543216789'
7     WHEN '+96144444444' THEN '123459876'
8     WHEN '+96155555555' THEN '567894321'
9     WHEN '+96166666666' THEN '678912345'
10    WHEN '+96177777777' THEN '876543219'
11    WHEN '+96188888888' THEN '345678912'
12    WHEN '+96199999999' THEN '456789123'
13    WHEN '+96112345678' THEN '234567891'
14    WHEN '+96124681357' THEN '789123456'
15    WHEN '+96165432198' THEN '654321987'
16    WHEN '+96111223344' THEN '987123456'
17    ELSE Employee_SSN
18
19 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:

```
1 BEGIN;
2
3 -- Step 1: Insert departments with Employee_SSN as NULL
4 INSERT INTO
5     Department (Name, Number_of_employees, Employee_SSN, Manager_start_date)
6 VALUES
7     ('Sales', 6, NULL, '2022-03-01'),
8     ('Marketing', 2, NULL, '2021-06-15'),
9     ('HR', 2, NULL, '2023-01-10'),
10    ('Finance', 3, NULL, '2019-10-05'),
11    ('Operations', 2, NULL, '2020-08-25'),
12    ('IT', 1, NULL, '2024-04-18'),
13    ('Customer Support', 2, NULL, '2022-09-12'),
14    ('Logistics', 3, NULL, '2020-11-20'),
15    ('Legal', 1, NULL, '2021-02-14'),
16    ('Training', 4, NULL, '2023-03-27'),
17    ('Facilities', 1, NULL, '2019-06-01'),
18    ('R&D', 1, NULL, '2022-05-10');
19
20 -- Step 2: Insert branches with Employee_SSN as NULL
21 INSERT INTO
22     Branch (Phone_number, Name, Country, State, City, Street, Building,
23             Apartment, Employee_SSN)
24 VALUES
25     ('+96111111111', 'Beirut Main', 'Lebanon', 'Beirut', 'Beirut', 'Hamra',
26      10, 12, NULL),
27     ('+96122222222', 'Tripoli Branch', 'Lebanon', 'North', 'Tripoli', 'Mina
28       Street', 5, 12, NULL),
29     ('+96133333333', 'Sidon Hub', 'Lebanon', 'South', 'Sidon', 'Corniche', 3,
30      6, NULL),
31     ('+96144444444', 'Zahle Branch', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St',
32      6, 4, NULL),
33     ('+96155555555', 'Jounieh Store', 'Lebanon', 'Mount Lebanon', 'Jounieh',
34       'Coastal Rd', 5, 3, NULL),
35     ('+96166666666', 'Byblos Outlet', 'Lebanon', 'Mount Lebanon', 'Byblos',
36       'Roman Street', 6, 7, NULL),
37     ('+96177777777', 'Tyre Shop', 'Lebanon', 'South', 'Tyre', 'Port Road', 2,
38      1, NULL),
39     ('+96188888888', 'Baalbek Point', 'Lebanon', 'Beqaa', 'Baalbek', 'Temple
40       Rd', 3, 2, NULL),
41     ('+96199999999', 'Batroun Corner', 'Lebanon', 'North', 'Batroun', 'Old
42       City', 1, 1, NULL),
43     ('+96112345678', 'Downtown Center', 'Lebanon', 'Beirut', 'Beirut', 'Beirut',
44       'Downtown', 9, 11, NULL),
```

```

34 ('+96124681357', 'Dora Warehouse', 'Lebanon', 'Mount Lebanon', 'Dora', 'Industrial Zone', 3, 10, NULL),
35 ('+96165432198', 'Aley Branch', 'Lebanon', 'Mount Lebanon', 'Aley', 'Souk Street', 6, 5, NULL),
36 ('+96111223344', 'Choueifat Station', 'Lebanon', 'Mount Lebanon', 'Choueifat', 'Railway Rd', 7, 3, NULL);
37
38 -- Step 3: Insert employees referencing Department_name
39 INSERT INTO
40 Employee (SSN, Position, Salary, Hire_date, Gender, Date_of_birth, Email,
41 First_name, Last_name, Phone_number, Country, State, City, Street,
42 Building, Apartment, Branch_phone_number, Supervisor_SSN,
43 Department_name)
44 VALUES
45 -- Sales Department Employees
46 ('123456789', 'Manager', 100000, '2019-01-15', 'Male', '1980-05-20', 'john.doe@example.com', 'John', 'Doe', '+96100000001', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 12, '+961111111111', NULL, 'Sales'),
47 ('1111111111', 'AdminAssistant', 50000, '2020-02-20', 'Female', '1990-08-15', 'jane.smith@example.com', 'Jane', 'Smith', '+96100000002', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 13, '+961111111111', 123456789, 'Sales'),
48 ('1111111112', 'Trainer', 60000, '2021-05-10', 'Male', '1985-11-30', 'peter.brown@example.com', 'Peter', 'Brown', '+96100000003', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 14, '+961111111111', 123456789, 'Sales'),
49 ('1111111113', 'ITSpecialist', 70000, '2021-07-01', 'Male', '1992-03-12', 'alex.jones@example.com', 'Alex', 'Jones', '+96100000004', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 15, '+961111111111', 123456789, 'Sales'),
50 ('1111111114', 'ProcurementOfficer', 55000, '2022-09-15', 'Female', '1995-06-20', 'lisa.green@example.com', 'Lisa', 'Green', '+96100000005', 'Lebanon', 'Beirut', 'Beirut', 'Hamra', 10, 16, '+961111111111', 123456789, 'Sales'),
51 ('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'),
52 -- Marketing Department Employees
53 ('987654321', 'MarketingHead', 90000, '2018-06-10', 'Female', '1982-09-25', 'sarah.johnson@example.com', 'Sarah', 'Johnson', '+96100000006', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5, 12, '+961222222222', NULL, 'Marketing'),
54 ('2222222222', 'AdminAssistant', 50000, '2021-08-01', 'Male', '1990-12-12', 'michael.taylor@example.com', 'Michael', 'Taylor', '+96100000007', 'Lebanon', 'North', 'Tripoli', 'Mina Street', 5, 13, '+961222222222', 987654321, 'Marketing'),
55 -- HR Department Employees

```

```

53 ('123459876', 'HRManager', 85000, '2023-01-10', 'Male', '1984-03-15', ,
54     david.wilson@example.com', 'David', 'Wilson', '+9610000008', 'Lebanon',
      ', 'Beqaa', 'Zahle', 'Beka St', 6, 4, '+96144444444', NULL, 'HR'),
55 ('333333333', 'AdminAssistant', 48000, '2023-02-15', 'Female', ,
56     1991-07-22', emily.davis@example.com', 'Emily', 'Davis', ,
57     '+9610000009', 'Lebanon', 'Beqaa', 'Zahle', 'Beka St', 6, 5, ,
58     '+96144444444', 123459876, 'HR'),
59 -- Finance Department Employees
60 ('567894321', 'FinanceManager', 95000, '2019-10-05', 'Female', ,
61     1978-11-08', laura.martin@example.com', 'Laura', 'Martin', ,
62     '+96100000010', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5,
63     3, '+96155555555', NULL, 'Finance'),
64 ('444444444', 'AdminAssistant', 47000, '2020-03-12', 'Male', '1989-04-18',
65     , daniel.moore@example.com', 'Daniel', 'Moore', '+96100000011', ,
66     Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5, 4, ,
67     '+96155555555', 567894321, 'Finance'),
68 ('555555555', 'ComplianceOfficer', 62000, '2021-07-23', 'Female', ,
69     1993-09-30', olivia.thomas@example.com', 'Olivia', 'Thomas', ,
70     '+96100000012', 'Lebanon', 'Mount Lebanon', 'Jounieh', 'Coastal Rd', 5,
71     5, '+96155555555', 567894321, 'Finance'),
72 -- Operations Department Employees
73 ('543216789', 'OperationsHead', 90000, '2018-08-15', 'Male', '1980-07-22',
74     , mark.stevens@example.com', 'Mark', 'Stevens', '+96100000013', ,
75     Lebanon', 'South', 'Sidon', 'Corniche', 3, 6, '+96133333333', NULL, ,
76     'Operations'),
77 ('666666666', 'AdminAssistant', 50000, '2019-05-20', 'Female', ,
78     1992-01-15', anna.miller@example.com', 'Anna', 'Miller', ,
79     '+96100000014', 'Lebanon', 'South', 'Sidon', 'Corniche', 3, 7, ,
80     '+96133333333', 543216789, 'Operations'),
81 -- IT Department Employee
82 ('678912345', 'ITSpecialist', 85000, '2020-03-10', 'Male', '1983-12-05',
83     , kevin.lee@example.com', 'Kevin', 'Lee', '+96100000015', 'Lebanon',
84     , Mount Lebanon', 'Byblos', 'Roman Street', 6, 7, '+96166666666', NULL,
85     , 'IT'),
86 -- Customer Support Department Employees
87 ('876543219', 'SupportManager', 80000, '2017-11-01', 'Female', ,
88     1985-04-18', laura.kim@example.com', 'Laura', 'Kim', '+96100000016',
89     , 'Lebanon', 'South', 'Tyre', 'Port Road', 2, 1, '+96177777777', NULL, ,
90     'Customer Support'),
91 ('777777777', 'AdminAssistant', 48000, '2019-06-15', 'Male', '1990-08-30',
92     , john.park@example.com', 'John', 'Park', '+96100000017', 'Lebanon',
93     , 'South', 'Tyre', 'Port Road', 2, 2, '+96177777777', 876543219, ,
94     'Customer Support'),
95 -- Logistics Department Employees
96 ('345678912', 'LogisticsHead', 90000, '2015-04-01', 'Male', '1979-02-25',
97     , peter.johnson@example.com', 'Peter', 'Johnson', '+96100000018', ,
98     Lebanon', 'Beqaa', 'Baalbek', 'Temple Rd', 3, 2, '+96188888888', NULL,
99     , 'Logistics'),

```

```

69 ('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'),
70 ('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'),
71 -- Legal Department Employee
72 ('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'),
73 -- Training Department Employees
74 ('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'),
75 ('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'),
76 ('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'),
77 ('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'),
78 -- Facilities Department Employee
79 ('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'),
80 -- R&D Department Employee
81 ('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'),
82 -- Driver Employees
83 ('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'),
84 ('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,

```

```

85     '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'),
86     ('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'),
87     ('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'),
88     ('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'),
89     ('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'),
90     ('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'),
91     ('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'),
92     ('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'),
93     ('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'),
94     ('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'),
95     ('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'),
96     ('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'),

```

```

97 ('023459876', 'Driver', 43000, '2015-05-22', 'Male', '1978-04-12', 'david
98 .lee@nexstore.io', 'David', 'Lee', '+96100000044', 'Lebanon', 'Beirut'
99 , 'Beirut', 'West Blvd', 5, 2, '+96111111111', NULL, 'Logistics');

100 -- Step 4: Update departments to set Employee_SSN (manager's SSN)
101 UPDATE Department
102 SET
103 Employee_SSN = CASE Name
104 WHEN 'Sales' THEN '123456789'
105 WHEN 'Marketing' THEN '987654321'
106 WHEN 'HR' THEN '123459876'
107 WHEN 'Finance' THEN '567894321'
108 WHEN 'Operations' THEN '543216789'
109 WHEN 'IT' THEN '678912345'
110 WHEN 'Customer Support' THEN '876543219'
111 WHEN 'Logistics' THEN '345678912'
112 WHEN 'Legal' THEN '456789123'
113 WHEN 'Training' THEN '789123456'
114 WHEN 'Facilities' THEN '987123456'
115 WHEN 'R&D' THEN '234567891'
116 ELSE Employee_SSN
117 END;

118 -- Step 5: Update branches to set Employee_SSN
119 UPDATE Branch
120 SET
121 Employee_SSN = CASE Phone_number
122 WHEN '+96111111111' THEN '123456789'
123 WHEN '+96122222222' THEN '987654321'
124 WHEN '+96133333333' THEN '543216789'
125 WHEN '+96144444444' THEN '123459876'
126 WHEN '+96155555555' THEN '567894321'
127 WHEN '+96166666666' THEN '678912345'
128 WHEN '+96177777777' THEN '876543219'
129 WHEN '+96188888888' THEN '345678912'
130 WHEN '+96199999999' THEN '456789123'
131 WHEN '+96112345678' THEN '234567891'
132 WHEN '+96124681357' THEN '789123456'
133 WHEN '+96165432198' THEN '654321987'
134 WHEN '+96111223344' THEN '987123456'
135 ELSE Employee_SSN
136 END;

137
138 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.

```
1 INSERT INTO
2     Driver (License_number, Driving_experience_years, License_expiry_date,
3             Employee_SSN)
4 VALUES
5     ('DL1001', 5, '2025-12-31', '023456789'),
6     ('DL1002', 3, '2026-06-30', '087654321'),
7     ('DL1003', 10, '2027-04-15', '067894321'),
8     ('DL1004', 7, '2028-01-10', '043216789'),
9     ('DL1005', 2, '2026-08-20', '078912345'),
10    ('DL1006', 6, '2025-11-25', '076543219'),
11    ('DL1007', 8, '2029-09-09', '045678912'),
12    ('DL1008', 12, '2025-05-05', '056789123'),
13    ('DL1009', 4, '2026-07-18', '034567891'),
14    ('DL1010', 15, '2030-03-30', '089123456'),
15    ('DL1011', 1, '2024-12-15', '054321987'),
16    ('DL1012', 9, '2027-02-14', '021987654'),
17    ('DL1013', 14, '2031-06-11', '087123456'),
18    ('DL1014', 11, '2029-12-21', '054789321'),
19    ('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.

```
1 INSERT INTO
2     Customer (Phone_number, Email, First_name, Last_name, Gender,
3                Registration_date, Password_hashed, Date_of_birth, Country, State,
4                City, Street, Building, Apartment)
5 VALUES
6     ('+96134567890', 'john.doe@gmail.com', 'John', 'Doe', 'Male', '2023-02-14',
7      '*****', '1990-03-05', 'Lebanon', 'Beirut', 'Beirut', 'Hamra',
8      12, 2),
9     ('+96198765432', 'jane.smith@yahoo.com', 'Jane', 'Smith', 'Female', '2022-06-12',
10    '*****', '1985-01-05', 'Lebanon', 'North', 'Tripoli',
11    'Mina Street', 4, 5),
12    ('+96156789014', 'alice.brown@outlook.com', 'Alice', 'Brown', 'Female', '2023-03-01',
13    '*****', '1992-01-10', 'Lebanon', 'South', 'Sidon',
14    'Corniche', 3, 6),
15    ('+96178901237', 'bob.jones@hotmail.com', 'Bob', 'Jones', 'Male', '2021-07-18',
16    '*****', '1995-04-15', 'Lebanon', 'Mount Lebanon',
17    'Jounieh', 'Coastal Rd', 6, 7),
18    ('+96167890123', 'charlie.evans@aol.com', 'Charlie', 'Evans', 'Male', '2022-05-30',
19    '*****', '1993-02-20', 'Lebanon', 'Beirut', 'Achrafieh',
20    'Armenia St', 9, 11),
```

```

9 ('+96189012348', 'diana.lee@icloud.com', 'Diana', 'Lee', 'Female', ' '
 2019-11-10', '*****', '1989-07-12', 'Lebanon', 'Beqaa', 'Zahle', ' '
 Beka St', 4, 3),
10 ('+96123456789', 'frank.wilson@protonmail.com', 'Frank', 'Wilson', 'Male'
 , '2024-01-01', '*****', '1993-01-17', 'Lebanon', 'Mount Lebanon',
 'Byblos', 'Roman Street', 5, 9),
11 ('+96121234569', 'emma.white@gmail.com', 'Emma', 'White', 'Female', ' '
 2023-11-15', '*****', '1985-12-25', 'Lebanon', 'South', 'Tyre', ' '
 Port Road', 2, 1),
12 ('+96156789012', 'george.king@live.com', 'George', 'King', 'Male', ' '
 2022-12-25', '*****', '2007-07-03', 'Lebanon', 'Mount Lebanon',
 'Antelias', 'Highway Rd', 8, 9),
13 ('+96178901234', 'jack.miller@yahoo.com', 'Jack', 'Miller', 'Male', ' '
 2024-02-09', '*****', '1992-04-17', 'Lebanon', 'Beqaa', 'Baalbek',
 'Temple Rd', 4, 3),
14 ('+96189012345', 'isabella.taylor@outlook.com', 'Isabella', 'Taylor', ' '
 Female', '2023-07-02', '*****', '1998-11-20', 'Lebanon', 'Beirut',
 'Downtown', 'Downtown', 3, 11),
15 ('+96121234567', 'kevin.harris@icloud.com', 'Kevin', 'Harris', 'Male', ' '
 2021-04-22', '*****', '1996-11-15', 'Lebanon', 'Beirut', 'Hamra', ' '
 Hamra', 4, 8),
16 ('+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.

```

1 INSERT INTO
2   Orders (Order_id, Notes, Payment_method, Total_amount, Is_online,
3           Employee_SSN, Customer_phone_number, Date, Driver_license_number)
4 VALUES
5   ('O101', 'Expedited delivery', 'Credit Card', 150, true, '123456789', ' '
6     '+96134567890', '2024-10-01', 'DL1001'),
7   ('O102', 'Gift wrap included', 'Cash', 200, false, '1111111111', ' '
8     '+96198765432', '2024-09-15', 'DL1002'),
9   ('O103', 'Deliver before 5 PM', 'PayPal', 75, true, '1111111111', ' '
10    '+96156789014', '2024-07-12', 'DL1003'),
11   ('O104', 'Call on arrival', 'Credit Card', 300, false, '1111111111', ' '
12    '+96178901237', '2024-04-07', 'DL1004'),
13   ('O105', 'Special instructions', 'Apple Pay', 500, true, '1111111111', ' '
14    '+96167890123', '2024-11-05', 'DL1005'),
15   ('O106', 'Holiday gift', 'Credit Card', 1000, true, '1111111112', ' '
16    '+96189012348', '2024-11-22', 'DL1006'),
17   ('O107', 'Contactless delivery', 'PayPal', 250, true, '1111111112', ' '
18    '+96123456789', '2024-07-12', 'DL1007'),

```

```

11 ('O108', 'Scheduled for 3 PM', 'Credit Card', 150, false, '111111112', '+96121234569', '2024-06-20', 'DL1008'),
12 ('O109', 'Express delivery', 'Cash', 500, false, '111111113', '+96156789012', '2024-11-09', 'DL1009'),
13 ('O110', 'New Years package', 'Apple Pay', 750, true, '111111113', '+96178901234', '2024-2-03', 'DL1010'),
14 ('O111', 'First-time discount', 'PayPal', 65, true, '111111113', '+96189012345', '2024-10-11', 'DL1011'),
15 ('O112', 'VIP priority', 'Credit Card', 800, true, '654321987', '+96121234567', '2024-07-14', 'DL1012'),
16 ('O113', 'Happy Birthday!', 'Apple Pay', 180, false, '654321987', '+96198765431', '2024-03-03', 'DL1013'),
17 ('O114', 'Clearance sale', 'PayPal', 450, false, '654321987', '+96134567890', '2024-11-05', 'DL1014'),
18 ('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.

```

1 INSERT INTO
2 Purchased (Product_SKU, Order_id, Quantity, Amount)
3 VALUES
4 (1001, 'O101', 2, 1200),
5 (1002, 'O102', 1, 1200),
6 (1003, 'O103', 1, 150),
7 (1004, 'O104', 2, 200),
8 (1005, 'O105', 1, 300),
9 (1006, 'O106', 1, 800),
10 (1007, 'O107', 1, 400),
11 (1008, 'O108', 3, 210),
12 (1009, 'O109', 5, 125),
13 (1010, 'O110', 2, 1000),
14 (1011, 'O111', 1, 50),
15 (1012, 'O112', 2, 60),
16 (1013, 'O113', 1, 150),
17 (1014, 'O114', 3, 120),
18 (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.

```
1 INSERT INTO
```

```

2   Reviews (Product_SKU, Customer_phone_number, Review_date, Rating, Comment
3       , Description)
4 VALUES
5   (1001, '+96134567890', '2024-09-01', 5, 'Great phone!', 'Excellent
6       performance'),
7   (1002, '+96198765432', '2024-08-20', 4, 'Good value', 'Worth the price'),
8   (1003, '+96156789014', '2024-07-10', 3, 'Comfortable chair', 'Could be
9       sturdier'),
10  (1004, '+96178901237', '2024-06-05', 5, 'Love these shoes', 'Perfect fit
11  '),
12  (1005, '+96167890123', '2024-05-15', 4, 'Nice sound', 'Great for
13  beginners'),
14  (1006, '+96189012348', '2024-04-22', 5, 'Amazing fridge', 'Lots of space
15  '),
16  (1007, '+96198765431', '2024-03-30', 4, 'Clear picture', 'Excellent for
17  gaming'),
18  (1008, '+96123456789', '2024-02-18', 3, 'Good blender', 'Noisy motor'),
19  (1009, '+96121234569', '2024-01-11', 5, 'Comfortable T-shirt', 'Soft
20  fabric'),
21  (1010, '+96156789012', '2024-09-25', 4, 'Nice smartwatch', 'Battery life
22  could be better'),
23  (1011, '+96178901234', '2024-08-05', 5, 'Informative book', 'Highly
24  recommended'),
25  (1012, '+96189012345', '2024-07-22', 4, 'Useful lamp', 'Bright and
26  adjustable'),
27  (1013, '+96198765431', '2024-06-12', 5, 'Excellent headphones', 'Superb
28  sound quality'),
29  (1014, '+96121234567', '2024-05-02', 4, 'Great soccer ball', 'Durable
30  material'),
31  (1015, '+96198765431', '2024-03-28', 5, 'Fantastic console', 'Best for
32  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.

```

1 INSERT INTO
2     Support_Ticket (Ticket_id, Description, Subject, Status, Priority,
3                     Employee_SSN, Customer_phone_number)
4 VALUES
5   ('T001', 'Issue with product delivery', 'Delivery Issue', 'Open', 'High',
6       '678912345', '+96134567890'),
7   ('T002', 'Request for refund', 'Refund Request', 'Closed', 'Medium', '
8       678912345', '+96198765432'),
9   ('T003', 'Product not working', 'Defective Product', 'Open', 'High', '
10      678912345', '+96156789014'),

```

```

7 ('T004', 'Inquiry about order status', 'Order Inquiry', 'Resolved', 'Low',
8     , '678912345', '+96178901237'),
9 ('T005', 'Delayed shipment', 'Shipment Delay', 'Open', 'Medium', '678912345', '+96167890123'),
10 ('T006', 'Cancel order request', 'Order Cancellation', 'Closed', 'Medium', '678912345', '+96189012348'),
11 ('T007', 'Issue with payment', 'Payment Issue', 'Resolved', 'High', '678912345', '+96123456789'),
12 ('T008', 'Exchange request', 'Product Exchange', 'Open', 'Medium', '678912345', '+96121234569'),
13 ('T009', 'Warranty inquiry', 'Warranty Inquiry', 'Resolved', 'Low', '678912345', '+96123456789'),
14 ('T010', 'Complaint about service', 'Service Complaint', 'Open', 'High', '678912345', '+96156789012'),
15 ('T011', 'Missing items in order', 'Missing Items', 'Open', 'Medium', '678912345', '+96189012345'),
16 ('T012', 'Subscription issue', 'Subscription Problem', 'Resolved', 'Low', '678912345', '+96178901237'),
17 ('T013', 'Wrong product delivered', 'Wrong Product', 'Open', 'High', '678912345', '+96167890123'),
18 ('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.

```

1 INSERT INTO
2     Wishlist (Product_SKU, Customer_phone_number, Total_amount)
3 VALUES
4     (1001, '+96134567890', 600),
5     (1002, '+96198765432', 1200),
6     (1003, '+96156789014', 150),
7     (1004, '+96178901237', 100),
8     (1005, '+96167890123', 300),
9     (1006, '+96189012348', 800),
10    (1007, '+96123456789', 400),
11    (1008, '+96121234569', 210),
12    (1009, '+96156789012', 125),
13    (1010, '+96178901234', 1000),
14    (1011, '+96189012345', 50),
15    (1012, '+96121234567', 60),
16    (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.

```
1 INSERT INTO
2   Working_hours (Branch_phone_number, Day, Opening_hour, Closing_hour)
3 VALUES
4   ('+96111111111', 'Monday', '09:00', '17:00'),
5   ('+96111111111', 'Tuesday', '09:00', '17:00'),
6   ('+96111111111', 'Wednesday', '09:00', '17:00'),
7   ('+96111111111', 'Thursday', '09:00', '17:00'),
8   ('+96111111111', 'Friday', '09:00', '14:00'),
9   ('+96111111111', 'Saturday', '09:00', '13:00'),
10  ('+96111111111', 'Sunday', NULL, NULL),
11  ('+96122222222', 'Monday', '08:00', '18:00'),
12  ('+96122222222', 'Tuesday', '08:00', '18:00'),
13  ('+96122222222', 'Wednesday', '08:00', '18:00'),
14  ('+96122222222', 'Thursday', '08:00', '18:00'),
15  ('+96122222222', 'Friday', '08:00', '13:00'),
16  ('+96122222222', 'Saturday', NULL, NULL),
17  ('+96122222222', 'Sunday', NULL, NULL),
18  ('+96133333333', 'Monday', '10:00', '19:00'),
19  ('+96133333333', 'Tuesday', '10:00', '19:00'),
20  ('+96133333333', 'Wednesday', '10:00', '19:00'),
21  ('+96133333333', 'Thursday', '10:00', '19:00'),
22  ('+96133333333', 'Friday', '10:00', '15:00'),
23  ('+96133333333', 'Saturday', '10:00', '14:00'),
24  ('+96133333333', 'Sunday', NULL, NULL),
25  ('+96144444444', 'Monday', '08:30', '17:30'),
26  ('+96144444444', 'Tuesday', '08:30', '17:30'),
27  ('+96144444444', 'Wednesday', '08:30', '17:30'),
28  ('+96144444444', 'Thursday', '08:30', '17:30'),
29  ('+96144444444', 'Friday', '08:30', '13:30'),
30  ('+96144444444', 'Saturday', NULL, NULL),
31  ('+96144444444', 'Sunday', NULL, NULL),
32  ('+96155555555', 'Monday', '09:00', '18:00'),
33  ('+96155555555', 'Tuesday', '09:00', '18:00'),
34  ('+96155555555', 'Wednesday', '09:00', '18:00'),
35  ('+96155555555', 'Thursday', '09:00', '18:00'),
36  ('+96155555555', 'Friday', '09:00', '14:00'),
37  ('+96155555555', 'Saturday', NULL, NULL),
38  ('+96155555555', 'Sunday', NULL, NULL),
39  ('+96166666666', 'Monday', '08:00', '20:00'),
40  ('+96166666666', 'Tuesday', '08:00', '20:00'),
41  ('+96166666666', 'Wednesday', '08:00', '20:00'),
42  ('+96166666666', 'Thursday', '08:00', '20:00'),
43  ('+96166666666', 'Friday', '08:00', '13:00'),
44  ('+96166666666', 'Saturday', '09:00', '14:00'),
45  ('+96166666666', 'Sunday', NULL, NULL),
```

```

46 ('+96177777777', 'Monday', '09:30', '18:30'),
47 ('+96177777777', 'Tuesday', '09:30', '18:30'),
48 ('+96177777777', 'Wednesday', '09:30', '18:30'),
49 ('+96177777777', 'Thursday', '09:30', '18:30'),
50 ('+96177777777', 'Friday', '09:30', '14:30'),
51 ('+96177777777', 'Saturday', NULL, NULL),
52 ('+96177777777', 'Sunday', NULL, NULL),
53 ('+96188888888', 'Monday', '09:00', '17:00'),
54 ('+96188888888', 'Tuesday', '09:00', '17:00'),
55 ('+96188888888', 'Wednesday', '09:00', '17:00'),
56 ('+96188888888', 'Thursday', '09:00', '17:00'),
57 ('+96188888888', 'Friday', '09:00', '13:00'),
58 ('+96188888888', 'Saturday', NULL, NULL),
59 ('+96188888888', 'Sunday', NULL, NULL),
60 ('+96199999999', 'Monday', '08:00', '19:00'),
61 ('+96199999999', 'Tuesday', '08:00', '19:00'),
62 ('+96199999999', 'Wednesday', '08:00', '19:00'),
63 ('+96199999999', 'Thursday', '08:00', '19:00'),
64 ('+96199999999', 'Friday', '08:00', '14:00'),
65 ('+96199999999', 'Saturday', '09:00', '13:00'),
66 ('+96199999999', 'Sunday', NULL, NULL),
67 ('+96112345678', 'Monday', '09:00', '18:00'),
68 ('+96112345678', 'Tuesday', '09:00', '18:00'),
69 ('+96112345678', 'Wednesday', '09:00', '18:00'),
70 ('+96112345678', 'Thursday', '09:00', '18:00'),
71 ('+96112345678', 'Friday', '09:00', '13:00'),
72 ('+96112345678', 'Saturday', NULL, NULL),
73 ('+96112345678', 'Sunday', NULL, NULL),
74 ('+96124681357', 'Monday', '08:30', '18:30'),
75 ('+96124681357', 'Tuesday', '08:30', '18:30'),
76 ('+96124681357', 'Wednesday', '08:30', '18:30'),
77 ('+96124681357', 'Thursday', '08:30', '18:30'),
78 ('+96124681357', 'Friday', '08:30', '14:30'),
79 ('+96124681357', 'Saturday', NULL, NULL),
80 ('+96124681357', 'Sunday', NULL, NULL),
81 ('+96165432198', 'Monday', '09:00', '19:00'),
82 ('+96165432198', 'Tuesday', '09:00', '19:00'),
83 ('+96165432198', 'Wednesday', '09:00', '19:00'),
84 ('+96165432198', 'Thursday', '09:00', '19:00'),
85 ('+96165432198', 'Friday', '09:00', '14:00'),
86 ('+96165432198', 'Saturday', '10:00', '14:00'),
87 ('+96165432198', 'Sunday', NULL, NULL),
88 ('+96111223344', 'Monday', '08:00', '17:00'),
89 ('+96111223344', 'Tuesday', '08:00', '17:00'),
90 ('+96111223344', 'Wednesday', '08:00', '17:00'),
91 ('+96111223344', 'Thursday', '08:00', '17:00'),
92 ('+96111223344', 'Friday', '08:00', '13:00'),
93 ('+96111223344', 'Saturday', NULL, NULL),

```

```
94 ('+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.

```
1 INSERT INTO
2     Dependent (Employee_SSN, Name, Gender, Date_of_birth, Relationship)
3 VALUES
4     ('123456789', 'Sarah Doe', 'Female', '2015-04-15', 'Daughter'),
5     ('987654321', 'James Smith', 'Male', '2013-07-20', 'Son'),
6     ('333333333', 'Emily Brown', 'Female', '2017-02-28', 'Daughter'),
7     ('333333333', 'Lucas Jones', 'Male', '2018-11-05', 'Son'),
8     ('444444444', 'Olivia Evans', 'Female', '2016-09-12', 'Daughter'),
9     ('543216789', 'Ethan White', 'Male', '2020-03-22', 'Son'),
10    ('666666666', 'Chloe Lee', 'Female', '2014-08-01', 'Daughter'),
11    ('876543219', 'Liam Harris', 'Male', '2015-12-15', 'Son'),
12    ('345678912', 'Mia Taylor', 'Female', '2018-10-03', 'Daughter'),
13    ('888888881', 'Noah Wilson', 'Male', '2021-06-08', 'Son'),
14    ('888888882', 'Sophia Martin', 'Female', '2019-05-25', 'Daughter'),
15    ('789123456', 'Benjamin Scott', 'Male', '2012-01-19', 'Son'),
16    ('999999991', 'Emma Anderson', 'Female', '2015-07-13', 'Daughter'),
17    ('999999992', 'Mason Miller', 'Male', '2018-03-09', 'Son'),
18    ('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.

```
1 INSERT INTO
2     Product_Image_URLs (Product_SKU, Product_Image_URL)
3 VALUES
4     ('1001', 'https://nextstore.io/images/product/1.jpg'),
5     ('1002', 'https://nextstore.io/images/product/2.jpg'),
6     ('1003', 'https://nextstore.io/images/product/3.jpg'),
7     ('1004', 'https://nextstore.io/images/product/4.jpg'),
8     ('1005', 'https://nextstore.io/images/product/5.jpg'),
9     ('1006', 'https://nextstore.io/images/product/6.jpg'),
10    ('1007', 'https://nextstore.io/images/product/7.jpg'),
11    ('1008', 'https://nextstore.io/images/product/8.jpg'),
12    ('1009', 'https://nextstore.io/images/product/9.jpg'),
13    ('1010', 'https://nextstore.io/images/product/10.jpg'),
14    ('1011', 'https://nextstore.io/images/product/11.jpg'),
15    ('1012', 'https://nextstore.io/images/product/12.jpg'),
16    ('1013', 'https://nextstore.io/images/product/13.jpg'),
```

```

17 ('1014', 'https://nextstore.io/images/product/14.jpg'),
18 ('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.

```

1 INSERT INTO
2   Review_Image_URLs (Product_SKU, Customer_phone_number, Product_Image_URL)
3 VALUES
4   ('1001', '+96134567890', 'https://nextstore.io/images/image1.jpg'),
5   ('1002', '+96198765432', 'https://nextstore.io/images/image2.jpg'),
6   ('1003', '+96156789014', 'https://nextstore.io/images/image3.jpg'),
7   ('1004', '+96178901237', 'https://nextstore.io/images/image4.jpg'),
8   ('1005', '+96167890123', 'https://nextstore.io/images/image5.jpg'),
9   ('1006', '+96189012348', 'https://nextstore.io/images/image6.jpg'),
10  ('1007', '+96123456789', 'https://nextstore.io/images/image7.jpg'),
11  ('1008', '+96121234569', 'https://nextstore.io/images/image8.jpg'),
12  ('1009', '+96156789012', 'https://nextstore.io/images/image9.jpg'),
13  ('1010', '+96178901234', 'https://nextstore.io/images/image10.jpg'),
14  ('1011', '+96189012345', 'https://nextstore.io/images/image11.jpg'),
15  ('1012', '+96121234567', 'https://nextstore.io/images/image12.jpg'),
16  ('1013', '+96198765431', 'https://nextstore.io/images/image13.jpg'),
17  ('1014', '+96198765432', 'https://nextstore.io/images/image14.jpg'),
18  ('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.

```

1 INSERT INTO
2   Located_in (Product_SKU, Branch_phone_number, Quantity, Shelf_location)
3 VALUES
4   ('1001', '+96111111111', 50, 'A1'),
5   ('1002', '+96122222222', 30, 'B2'),
6   ('1003', '+96133333333', 20, 'C3'),
7   ('1004', '+96144444444', 15, 'D4'),
8   ('1005', '+96155555555', 60, 'E5'),
9   ('1006', '+96166666666', 25, 'F6'),
10  ('1007', '+96177777777', 10, 'G7'),
11  ('1008', '+96188888888', 35, 'H8'),
12  ('1009', '+96199999999', 40, 'I9'),
13  ('1010', '+96112345678', 50, 'J10'),
14  ('1011', '+96124681357', 70, 'K11'),
15  ('1012', '+96124681357', 20, 'L12'),

```

```
16 ('1013', '+96165432198', 15, 'M13'),  
17 ('1014', '+96165432198', 5, 'N14'),  
18 ('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.

```
1   INSERT INTO
2     Colors (Product_SKU, Product_color)
3
4   VALUES
5
6     ('1001', 'Red'),
7     ('1002', 'Blue'),
8     ('1003', 'Green'),
9     ('1004', 'Black'),
10    ('1005', 'White'),
11    ('1006', 'Yellow'),
12    ('1007', 'Pink'),
13    ('1008', 'Purple'),
14    ('1009', 'Orange'),
15    ('1010', 'Brown'),
16    ('1011', 'Gray'),
17    ('1012', 'Gold'),
18    ('1013', 'Silver'),
19    ('1014', 'Beige'),
20    ('1015', 'Navy');
```

Code Snippet 11.46 : Insert into Colors Table

11.1.3.18 Coupon

The following script inserts data into the Coupon table.

```
1 INSERT INTO
2   Coupon (Code, Description, Discount_percent, Times_used,
3           Minimum_order_amount, Maximum_order_amount, Usage_limit, Valid_from,
4           Valid_to, Order_ID, Discount_amount, Redeem_date)
5 VALUES
6   ('DIS10', '10% off', 10, 25, 50, 500, 100, '2024-01-01', '2024-12-31', '0101', 5, '2024-10-01'),
7   ('DIS20', '20% off', 20, 40, 100, 1000, 75, '2024-01-01', '2024-12-31', '0102', 10, '2024-09-15'),
8   ('SAVE15', '15% off', 15, 50, 200, 1000, 25, '2024-01-01', '2024-12-31', '0104', 15, '2024-07-07'),
9   ('BOGO', 'Buy 1 Get 1', 50, 20, 100, 500, 50, '2024-01-01', '2024-12-31', '0105', 20, '2024-06-06'),
```

```

8 ('HOLIDAY50', '50% off', 50, 15, 200, 1000, 30, '2024-01-01', '2024-08-01
  , 'O106', 50, '2024-04-01'),
9 ('FLASH5', '5% off', 5, 150, 25, 250, 50, '2024-01-01', '2024-12-31', 'O107',
  2, '2024-07-20'),
10 ('SUMMER30', '30% off', 30, 80, 150, 1500, 30, '2024-01-01', '2024-12-31',
   , 'O108', 45, '2024-09-20'),
11 ('BLCKFRIDAY', '40% off', 40, 100, 300, 2000, 75, '2024-01-01', '2024-12-31',
   , 'O109', 60, '2024-11-01'),
12 ('NEWYEAR25', '25% off', 25, 70, 100, 1000, 50, '2024-01-01', '2024-12-31',
   , 'O110', 50, '2024-10-31'),
13 ('WELCOME', '10% for new users', 10, 10, 50, 500, 25, '2024-01-01', '2024-12-31',
   , 'O111', 5, '2024-11-15'),
14 ('VIP20', '20% VIP discount', 20, 30, 150, 1500, 50, '2024-01-01', '2024-12-31',
   , 'O112', 30, '2024-05-01'),
15 ('BIRTHDAY', '25% off on birthday', 25, 20, 100, 1000, 25, '2024-01-01',
   , '2024-12-31', 'O113', 10, '2024-04-10'),
16 ('CLEARANCE', 'Up to 70% off', 70, 50, 500, 5000, 100, '2024-01-01', '2024-11-15',
   , 'O114', 350, '2024-05-14'),
17 ('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.

```

1 INSERT INTO
2   Department_location (Department_name, Location)
3 VALUES
4   ('Sales', 'Beirut'),
5   ('Marketing', 'Tripoli'),
6   ('HR', 'Zahle'),
7   ('Finance', 'Jounieh'),
8   ('Operations', 'Sidon'),
9   ('IT', 'Byblos'),
10  ('Customer Support', 'Tyre'),
11  ('Logistics', 'Baalbek'),
12  ('Legal', 'Batroun'),
13  ('Training', 'Dora'),
14  ('Facilities', 'Antelias'),
15  ('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.

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

Code Snippet 11.49 : *Select from Supplier Table*

	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.

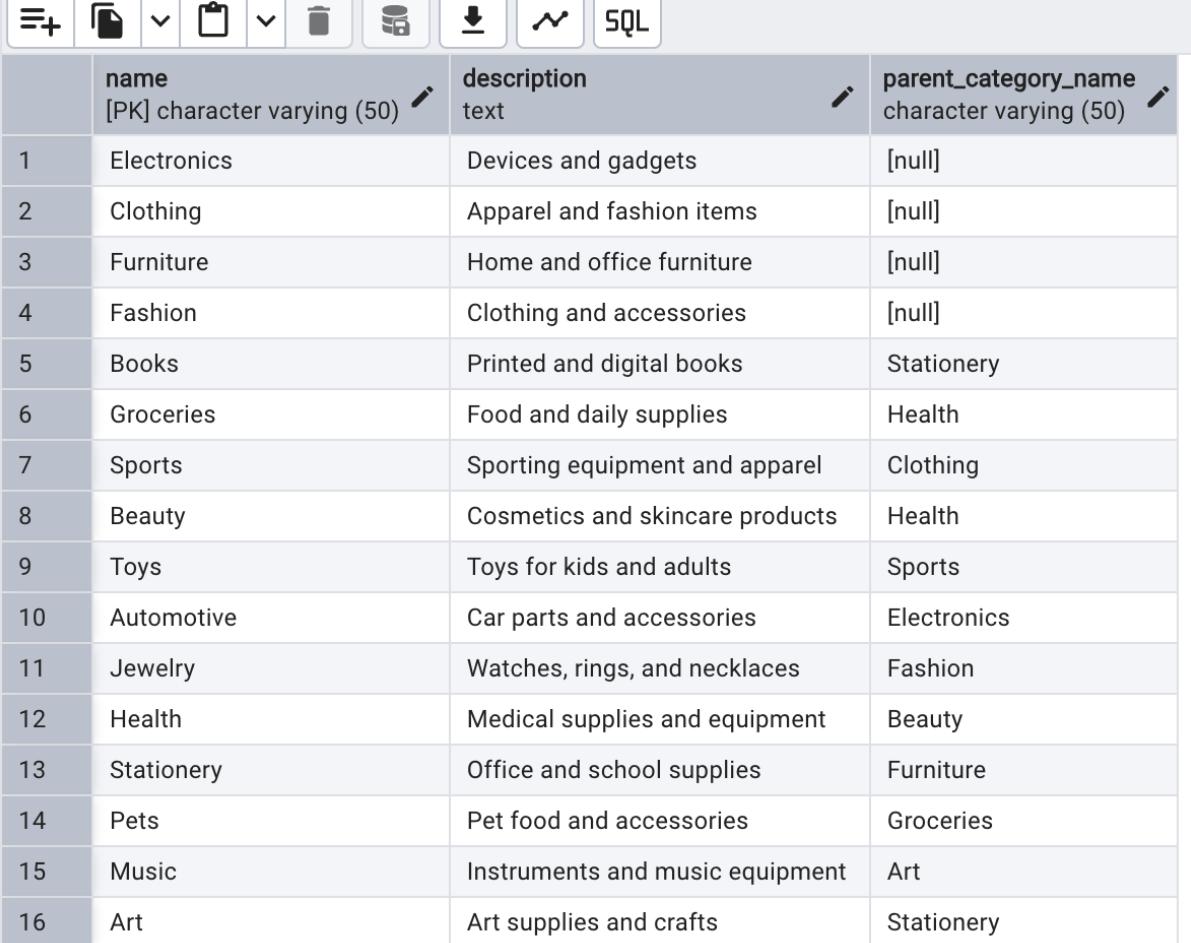
```
1 SELECT * FROM Category;
```

Code Snippet 11.50 : *Select from Category Table*

Query Query History

1 `SELECT * FROM Category;`

Data Output Messages Notifications



The screenshot shows a database interface with a toolbar at the top containing icons for new table, file operations, and SQL. Below the toolbar is a table titled 'Category' with 16 rows. The table has three columns: 'name' (PK), 'description', and 'parent_category_name'. The data is as follows:

	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

The screenshot shows a database interface with a toolbar at the top. Below the toolbar is a table header with columns: sku, name, price, description, category_name, supplier_website, weight, brand, width, height, and length. The table contains 15 rows of product data. The data includes items like a Smartphone, Laptop, Office Chair, Running Shoes, Acoustic Guitar, Refrigerator, LED TV, Blender, T-Shirt, Watch, Textbook, Desk Lamp, Wireless Headphones, Soccer Ball, and a Gaming Console. Each row provides details such as the product name, price, description, category, supplier website, dimensions, and brand.

	sku	name	price	description	category_name	supplier_website	weight	brand	width	height	length
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.sportswear.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

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

Data Output Messages Notifications

The screenshot shows a database interface with a toolbar at the top. Below the toolbar is a table header with columns: phone_number, name, country, state, city, street, building, apartment, and employee_ssn. The table contains 13 rows of branch data. The data includes branches like Beirut Main, Tripoli Branch, Sidon Hub, Zahle Branch, Jounieh Store, Byblos Outlet, Tyre Shop, Baalbek Point, Batroun Corner, Downtown Center, Dora Warehouse, Aley Branch, and Choueifat Station. Each row provides details such as the branch name, location, and contact information.

	phone_number	name	country	state	city	street	building	apartment	employee_ssn
1	+9611111111	Beirut Main	Lebanon	Beirut	Beirut	Hamra	10	12	123456789
2	+9612222222	Tripoli Branch	Lebanon	North	Tripoli	Mina Street	5	12	987654321
3	+9613333333	Sidon Hub	Lebanon	South	Sidon	Corniche	3	6	543216789
4	+9614444444	Zahle Branch	Lebanon	Beqaa	Zahle	Beka St	6	4	123459876
5	+9615555555	Jounieh Store	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	5	3	567894321
6	+9616666666	Byblos Outlet	Lebanon	Mount Lebanon	Byblos	Roman Street	6	7	678912345
7	+9617777777	Tyre Shop	Lebanon	South	Tyre	Port Road	2	1	876543219
8	+9618888888	Baalbek Point	Lebanon	Beqaa	Baalbek	Temple Rd	3	2	345678912
9	+9619999999	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

	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*

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

Query Query History

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

Data Output Messages Notifications

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

	phone_number [PK] character varying (50)	email character varying (50)	first_name character varying (50)	last_name character varying (50)	gender character varying (50)	registration_date date	password_hashed character varying (50)	date_of_birth date	country character varying (50)	state character varying (50)	city character varying (50)	street character varying (50)	building integer	apartment integer
1	+96134567890	john.doe@gmail.com	John	Doe	Male	2023-02-14	*****	1990-03-05	Lebanon	Beruit	Beruit	Hamra	12	2
2	+96198765432	jane.smith@yahoo.com	Jane	Smith	Female	2022-06-12	*****	1985-01-05	Lebanon	North	Tripoli	Mina Street	4	5
3	+96136789014	alice.brown@outlook.com	Alice	Brown	Female	2023-03-01	*****	1992-01-10	Lebanon	South	Sidon	Comiche	3	6
4	+96178901237	bob.jones@hotmail.com	Bob	Jones	Male	2021-07-18	*****	1995-04-15	Lebanon	Mount Lebanon	Jounieh	Coastal Rd	6	7
5	+96167890123	charlie.evans@aol.com	Charlie	Evans	Male	2022-05-30	*****	1993-02-20	Lebanon	Beruit	Achrafieh	Armenia St	9	11
6	+96189012348	diana.lee@icloud.com	Diana	Lee	Female	2019-11-10	*****	1989-07-12	Lebanon	Bekaa	Zahré	Beka St	4	3
7	+96123456789	frank.wilson@protonmail.com	Frank	Wilson	Male	2024-01-01	*****	1993-01-17	Lebanon	Mount Lebanon	Byblos	Roman Street	5	9
8	+96121234569	emma.white@gmail.com	Emma	White	Female	2023-11-15	*****	1985-12-25	Lebanon	South	Tyre	Port Road	2	1
9	+96156789012	george.king@yahoo.com	George	King	Male	2022-12-25	*****	2007-07-03	Lebanon	Mount Lebanon	Antelias	Highway Rd	8	9
10	+96178901234	jack.miller@yahoo.com	Jack	Miller	Male	2024-02-09	*****	1992-04-17	Lebanon	Bekaa	Baabda	Temple Rd	4	3
11	+96189012345	isabella.taylor@outlook.com	Isabella	Taylor	Female	2023-07-02	*****	1998-11-20	Lebanon	Beruit	Downtown	Downtown	3	11
12	+96121234567	kevin.harris@icloud.com	Kevin	Harris	Male	2021-04-22	*****	1996-11-15	Lebanon	Beruit	Hamra	Hamra	4	8
13	+96198765431	mike.anderson@protonmail.com	Mike	Anderson	Male	2023-05-03	*****	1998-01-11	Lebanon	Mount Lebanon	Choueifat	Railway Rd	7	3

Figure 11.8: Select from Customer Table

11.1.4.9 Orders

The following script selects data from the Orders table.

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

Code Snippet 11.57 : Select from Orders Table

	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.

```
1 SELECT * FROM Purchased;
```

Code Snippet 11.58 : Select from Purchased Table

Query Query History

```
1   SELECT * FROM Purchased;
```

Data Output Messages Notifications



	product_sku [PK] character varying (50)	order_id [PK] character (10)	quantity integer	amount numeric
1	1001	0101	2	1200
2	1002	0102	1	1200
3	1003	0103	1	150
4	1004	0104	2	200
5	1005	0105	1	300
6	1006	0106	1	800
7	1007	0107	1	400
8	1008	0108	3	210
9	1009	0109	5	125
10	1010	0110	2	1000
11	1011	0111	1	50
12	1012	0112	2	60
13	1013	0113	1	150
14	1014	0114	3	120
15	1015	0115	1	500

Figure 11.10: *Select from Purchased Table*

11.1.4.11 Reviews

The following script selects data from the Reviews table.

```
1   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: 1

	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.

```
1 SELECT * FROM Wishlist;
```

Code Snippet 11.61 : Select from Wishlist Table

The screenshot shows a database interface with a query editor and a results grid. The query editor contains the SQL command: `SELECT * FROM Wishlist;`. The results grid displays 13 rows of data with columns: product_sku, customer_phone_number, and total_amount.

	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.

```
1 SELECT * FROM Working_Hours;
```

Code Snippet 11.62 : Select from Working Hours Table

Query Query History

```
1  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.

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

Code Snippet 11.63 : Select from Dependent Table

Query Query History

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

Data Output Messages Notifications

SQL

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

The screenshot shows a database interface with a toolbar at the top containing icons for file operations, a refresh button, and a SQL tab. Below the toolbar is a table with 15 rows of data. The table has five columns: product_sku, branch_phone_number, quantity, and shelf_location. The data is as follows:

	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.

```
1 | SELECT * FROM Coupon;
```

Code Snippet 11.68 : Select from Coupon Table

Query Query History

1 | `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	0101	5	2024-10-01
2	DIS20	20% off	20	40	100	1000	75	2024-01-01	2024-12-31	0102	10	2024-09-15
3	SAVE15	15% off	15	50	200	1000	25	2024-01-01	2024-12-31	0104	15	2024-07-07
4	BOGO	Buy 1 Get 1	50	20	100	500	50	2024-01-01	2024-12-31	0105	20	2024-06-06
5	HOLIDAY50	50% off	50	15	200	1000	30	2024-01-01	2024-08-01	0106	50	2024-04-01
6	FLASH5	5% off	5	150	25	250	50	2024-01-01	2024-12-31	0107	2	2024-07-20
7	SUMMER30	30% off	30	80	150	1500	30	2024-01-01	2024-12-31	0108	45	2024-09-20
8	BLOCKFRIDAY	40% off	40	100	300	2000	75	2024-01-01	2024-12-31	0109	60	2024-11-01
9	NEWYEAR25	25% off	25	70	100	1000	50	2024-01-01	2024-12-31	0110	50	2024-10-31
10	WELCOME	10% for new users	10	10	50	500	25	2024-01-01	2024-12-31	0111	5	2024-11-15
11	VIP20	20% VIP discount	20	30	150	1500	50	2024-01-01	2024-12-31	0112	30	2024-05-01
12	BIRTHDAY	25% off on birthday	25	20	100	1000	25	2024-01-01	2024-12-31	0113	10	2024-04-10
13	CLEARANCE	Up to 70% off	70	50	500	5000	100	2024-01-01	2024-11-15	0114	350	2024-05-14
14	LOYALTY	15% off for loyal customers	15	15	50	1000	60	2024-01-01	2024-12-31	0115	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.

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

Code Snippet 11.69 : Select from Department Location Table

The screenshot shows a SQL query interface with the following details:

- Query History:** The tab is labeled "Query History".
- Query:** The code is `SELECT * FROM Department_Location;`.
- Data Output:** The tab is labeled "Data Output".
- Table Headers:** The table has two columns: "department_name" [PK] character varying (50) and "location" [PK] character varying (50).
- Table Data:** The table contains 12 rows of data:

	department_name	location
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

```

1  SELECT
2      Customer.First_name,
3      Customer.Last_name,
4      COUNT(Orders.Order_id) AS Total_Orders
5  FROM
6      Customer

```

```

7   JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
8 GROUP BY
9   Customer.First_name,
10  Customer.Last_name
11 ORDER BY
12  Total_Orders DESC
13 FETCH FIRST
14  5 ROWS ONLY;

```

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

The screenshot shows a SQL query editor interface. At the top, there are tabs for "Query" and "Query History", with "Query" currently selected. Below the tabs is the SQL code for finding the top 5 customers with the highest number of orders. The code uses a JOIN clause to link the Customer and Orders tables based on their phone numbers, groups the results by customer first and last names, counts the total number of orders for each customer, orders the results by total orders in descending order, and finally limits the output to the top 5 rows.

```

1 < SELECT
2   Customer.First_name,
3   Customer.Last_name,
4   COUNT(Orders.Order_id) AS Total_Orders
5 FROM
6   Customer
7   JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
8 GROUP BY
9   Customer.First_name,
10  Customer.Last_name
11 ORDER BY
12  Total_Orders DESC
13 FETCH FIRST
14  5 ROWS ONLY;

```

Below the query area, there are tabs for "Data Output", "Messages", and "Notifications", with "Data Output" selected. The data output section displays a table with five rows, showing the first name, last name, and total number of orders for each customer. The columns are labeled "first_name", "last_name", and "total_orders". The data shows five customers: Isabella Taylor (2 orders), John Doe (2 orders), Diana Lee (1 order), Emma White (1 order), and Bob Jones (1 order).

	first_name	last_name	total_orders
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

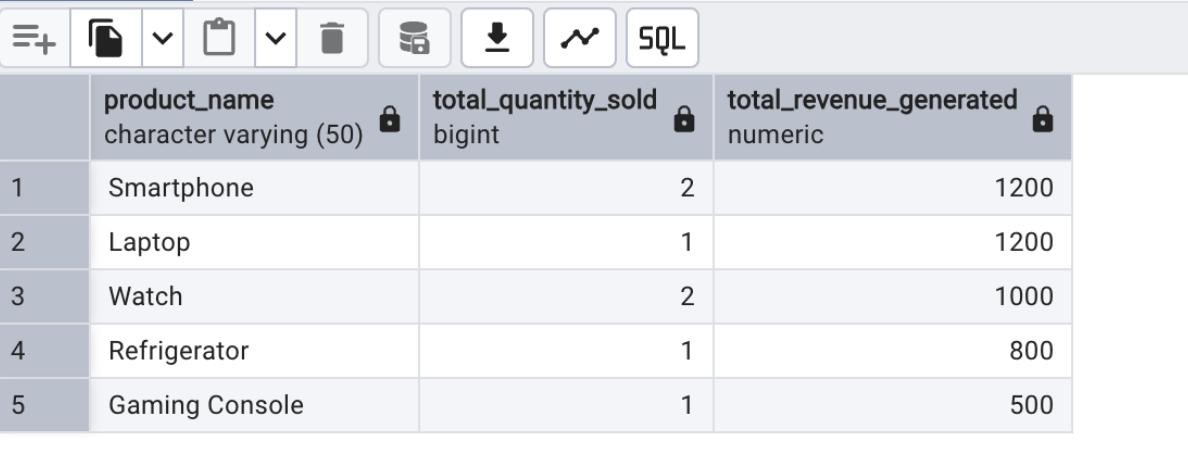
```
1 SELECT
2     Product.Name AS Product_Name,
3     SUM(Purchased.Quantity) AS Total_Quantity_Sold,
4     SUM(Purchased.Amount) AS Total_Revenue_Generated
5 FROM
6     Product,
7     Purchased
8 WHERE
9     Product.SKU = Purchased.Product_SKU
10 GROUP BY
11     Product.Name
12 ORDER BY
13     Total_Revenue_Generated DESC
14 FETCH FIRST
15     5 ROWS ONLY;
```

Code Snippet 11.71 : *Products with Highest Revenue*

Query Query History

```
1 ▾ SELECT
2     Product.Name AS Product_Name,
3     SUM(Purchased.Quantity) AS Total_Quantity_Sold,
4     SUM(Purchased.Amount) AS Total_Revenue_Generated
5 FROM
6     Product,
7     Purchased
8 WHERE
9     Product.SKU = Purchased.Product_SKU
10 GROUP BY
11     Product.Name
12 ORDER BY
13     Total_Revenue_Generated DESC
14 FETCH FIRST
15     5 ROWS ONLY;
```

Data Output Messages Notifications



The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons for file operations like new, save, and copy. The main area displays a results grid for the executed SQL query. The grid has three columns: 'product_name' (character varying (50)), 'total_quantity_sold' (bigint), and 'total_revenue_generated' (numeric). The data shows five rows of results:

	product_name	total_quantity_sold	total_revenue_generated
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

```
1 WITH
2     Product_Sales AS (
```

```

3   SELECT
4     L.Branch_phone_number,
5     P.Product_SKU,
6     SUM(P.Quantity) AS Total_Sold
7   FROM
8     Purchased P
9     JOIN Located_in L ON P.Product_SKU = L.Product_SKU
10    GROUP BY
11      L.Branch_phone_number,
12      P.Product_SKU
13  ) ,
14  Max_Sales_Per_Branch AS (
15    SELECT
16      Branch_phone_number,
17      MAX(Total_Sold) AS Max_Sold
18    FROM
19      Product_Sales
20    GROUP BY
21      Branch_phone_number
22  )
23  SELECT
24    PS.Branch_phone_number,
25    PS.Product_SKU,
26    PS.Total_Sold
27  FROM
28    Product_Sales PS
29    JOIN Max_Sales_Per_Branch MSPB ON PS.Branch_phone_number = MSPB.
30      Branch_phone_number
31      AND PS.Total_Sold = MSPB.Max_Sold;

```

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

Query Query History

```

1 ✓ WITH Product_Sales AS (
2     SELECT
3         L.Branch_phone_number,
4         P.Product_SKU,
5         SUM(P.Quantity) AS Total_Sold
6     FROM
7         Purchased P
8     JOIN
9         Located_in L ON P.Product_SKU = L.Product_SKU
10    GROUP BY
11        L.Branch_phone_number, P.Product_SKU
12    ),
13    Max_Sales_Per_Branch AS (
14        SELECT
15            Branch_phone_number,
16            MAX(Total_Sold) AS Max_Sold
17        FROM
18            Product_Sales
19        GROUP BY
20            Branch_phone_number
21    )
22    SELECT
23        PS.Branch_phone_number,
24        PS.Product_SKU,
25        PS.Total_Sold
26    FROM
27        Product_Sales PS
28    JOIN
29        Max_Sales_Per_Branch MSPB
30        ON PS.Branch_phone_number = MSPB.Branch_phone_number
31        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

```
1 SELECT
2     Customer.Phone_number,
3     Customer.First_name,
4     Customer.Last_name,
5     SUM(Purchased.Amount) AS Lifetime_Value
6 FROM
7     Customer
8     JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
9     JOIN Purchased ON Orders.Order_id = Purchased.Order_id
10 GROUP BY
11     Customer.Phone_number,
12     Customer.First_name,
13     Customer.Last_name
14 ORDER BY
15     Lifetime_Value DESC;
```

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

Query History

```

1  SELECT
2      Customer.Phone_number,
3      Customer.First_name,
4      Customer.Last_name,
5      SUM(Purchased.Amount) AS Lifetime_Value
6  FROM
7      Customer
8      JOIN Orders ON Customer.Phone_number = Orders.Customer_phone_number
9      JOIN Purchased ON Orders.Order_id = Purchased.Order_id
10 GROUP BY
11     Customer.Phone_number,
12     Customer.First_name,
13     Customer.Last_name
14 ORDER BY
15     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

```
1 SELECT
2     Product.SKU,
3     Product.Name,
4     SUM(Purchased.Quantity) AS Total_Sold,
5     Product.Price * SUM(Purchased.Quantity) AS Total_Revenue,
6     COUNT(DISTINCT Purchased.Order_id) AS Total_Orders
7 FROM
8     Product
9     LEFT JOIN Purchased ON Product.SKU = Purchased.Product_SKU
10 GROUP BY
11     Product.SKU,
12     Product.Name,
13     Product.Price
14 HAVING
15     SUM(Purchased.Quantity) < 10
16     OR Product.Price * SUM(Purchased.Quantity) < 500;
```

Code Snippet 11.74 : *Underperforming Products*

Query Query History

```

1  SELECT
2      Product.SKU,
3      Product.Name,
4      SUM(Purchased.Quantity) AS Total_Sold,
5      Product.Price * SUM(Purchased.Quantity) AS Total_Revenue,
6      COUNT(DISTINCT Purchased.Order_id) AS Total_Orders
7  FROM
8      Product
9      LEFT JOIN Purchased ON Product.SKU = Purchased.Product_SKU
10 GROUP BY
11     Product.SKU,
12     Product.Name,
13     Product.Price
14 HAVING
15     SUM(Purchased.Quantity) < 10
16     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

```
1 WITH
2   ProductPairs AS (
3     SELECT
4       p1.Product_SKU AS Product_1,
5       p2.Product_SKU AS Product_2,
6       COUNT(*) AS Pair_Count
7     FROM
8       Purchased p1
9     JOIN Purchased p2 ON p1.Order_id = p2.Order_id
10    AND p1.Product_SKU < p2.Product_SKU
11   GROUP BY
12     p1.Product_SKU,
13     p2.Product_SKU
14   )
15  SELECT
16    Product_1,
17    Product_2,
18    Pair_Count
19  FROM
20    ProductPairs
21 WHERE
22  Pair_Count > 5
23 ORDER BY
24  Pair_Count DESC;
```

Code Snippet 11.75 : *Find Popular Products Combinations*

Query Query History

```
1 < WITH
2     ProductPairs AS (
3         SELECT
4             p1.Product_SKU AS Product_1,
5             p2.Product_SKU AS Product_2,
6             COUNT(*) AS Pair_Count
7         FROM
8             Purchased p1
9         JOIN Purchased p2 ON p1.Order_id = p2.Order_id
10        AND p1.Product_SKU < p2.Product_SKU
11        GROUP BY
12            p1.Product_SKU,
13            p2.Product_SKU
14        )
15        SELECT
16            Product_1,
17            Product_2,
18            Pair_Count
19        FROM
20            ProductPairs
21        WHERE
22            Pair_Count > 5
23        ORDER BY
24            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

```
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;
```

Code Snippet 11.76 : *Products with the Most Discounts*

The screenshot shows a SQL query editor interface. At the top, there are tabs for "Query" (which is selected) and "Query History". Below the tabs is the SQL code for the query:

```

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;

```

Below the code is a toolbar with icons for file operations (New, Open, Save, Print, Copy, Paste, Find, Refresh, SQL), and a "Show" dropdown menu.

The main area displays the "Data Output" tab, which contains the results of the query:

	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

```

1  SELECT
2      EXTRACT (
3          MONTH
4      FROM
5          Orders.Date
6      ) AS Month,
7      Product.Category_name,
8      SUM(Purchased.Quantity) AS Total_Sold
9  FROM
10     Orders
11    JOIN Purchased ON Orders.Order_id = Purchased.Order_id
12    JOIN Product ON Purchased.Product_SKU = Product.SKU
13 GROUP BY
14     EXTRACT (
15         MONTH
16         FROM
17             Orders.Date
18     ) ,
19     Product.Category_name
20 ORDER BY
21     Month,
22     Total_Sold DESC;

```

Code Snippet 11.77 : Seasonal Trends for Product Categories

The screenshot shows a SQL query editor interface. At the top, there are tabs for 'Query' (which is selected), 'Query History', and other unlabelled tabs. Below the tabs is the SQL code for the query:

```

1  SELECT
2      EXTRACT(
3          MONTH
4      FROM
5          Orders.Date
6      ) AS Month,
7      Product.Category_name,
8      SUM(Purchased.Quantity) AS Total_Sold
9  FROM
10     Orders
11    JOIN Purchased ON Orders.Order_id = Purchased.Order_id
12    JOIN Product ON Purchased.Product_SKU = Product.SKU
13 GROUP BY
14     EXTRACT(
15         MONTH
16     FROM
17         Orders.Date
18     ),
19     Product.Category_name
20 ORDER BY
21     Month,
22     Total_Sold DESC;

```

Below the code is a 'Data Output' section containing a table with the results of the query. The table has three tabs above it: 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected. The table has three columns: 'month' (numeric), 'category_name' (character varying (50)), and 'total_sold' (bigint). The data is as follows:

	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

```

1  WITH
2      StockLevels AS (
3          SELECT
4              Branch_phone_number,
5              Product_SKU,
6              Quantity,
7              ROW_NUMBER() OVER (
8                  PARTITION BY
9                      Branch_phone_number
10                 ORDER BY
11                     Quantity ASC
12             ) AS Stock_Rank
13         FROM
14             Located_in
15     )
16     SELECT
17         Branch.Phone_number AS Branch_ID,
18         Branch.Name AS Branch_Name,
19         StockLevels.Product_SKU,
20         StockLevels.Quantity AS Current_Stock
21     FROM
22         Branch
23         JOIN StockLevels ON Branch.Phone_number = StockLevels.Branch_phone_number
24     WHERE
25         StockLevels.Stock_Rank <= 3
26     ORDER BY
27         Branch_ID,
28         Current_Stock ASC;

```

Code Snippet 11.78 : *Branches with Stock Shortages*

Query Query History

```

1  WITH
2      StockLevels AS (
3          SELECT
4              Branch_phone_number,
5              Product_SKU,
6              Quantity,
7              ROW_NUMBER() OVER (
8                  PARTITION BY
9                      Branch_phone_number
10                 ORDER BY
11                     Quantity ASC
12             ) AS Stock_Rank
13         FROM
14             Located_in
15     )
16
SELECT
17     Branch.Phone_number AS Branch_ID,
18     Branch.Name AS Branch_Name,
19     StockLevels.Product_SKU,
20     StockLevels.Quantity AS Current_Stock
21
FROM
22     Branch
23     JOIN StockLevels ON Branch.Phone_number = StockLevels.Branch_phone_number
24
WHERE
25     StockLevels.Stock_Rank <= 3
26
ORDER BY
27     Branch_ID,
28     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 \leq 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

```

1   WITH
2     BranchSales AS (
3       SELECT
4         Branch_phone_number,
5         SUM(Purchased.Amount) AS Total_Revenue
6       FROM
7         Purchased
8         JOIN Located_in ON Purchased.Product_SKU = Located_in.Product_SKU
9       GROUP BY
10      Branch_phone_number
11    )
12   SELECT
13     Branch_phone_number,
14     Total_Revenue
15   FROM
16   (
17     SELECT
18       Branch_phone_number,
19       Total_Revenue,
20       RANK() OVER (
21         ORDER BY
22           Total_Revenue DESC
23       ) AS Rank
24     FROM
25       BranchSales
26   ) RankedSales
27 WHERE
28   Rank = 1;

```

Code Snippet 11.79 : Best Performing Branch

Query Query History

```

1  WITH
2      BranchSales AS (
3          SELECT
4              Branch_phone_number,
5                  SUM(Purchased.Amount) AS Total_Revenue
6          FROM
7              Purchased
8                  JOIN Located_in ON Purchased.Product_SKU = Located_in.Product_SKU
9          GROUP BY
10             Branch_phone_number
11     )
12     SELECT
13         Branch_phone_number,
14             Total_Revenue
15     FROM
16     (
17         SELECT
18             Branch_phone_number,
19                 Total_Revenue,
20                 RANK() OVER (
21                     ORDER BY
22                         Total_Revenue DESC
23                 ) AS Rank
24     FROM
25         BranchSales
26     ) RankedSales
27     WHERE
28         Rank = 1;

```

Data Output Messages Notifications

	branch_phone_number	total_revenue
character varying (25)	1	1200
2	+9611111111	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

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

Code Snippet 11.80 : *Customer Orders View*

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 rows 1-13 of 13

	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

```

```
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;
```

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 to 12

```

	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

```

```

4 Product.SKU,
5 Product.Name,
6 SUM(Purchased.Amount) AS Total_Revenue
7 FROM
8 Product
9 JOIN Purchased ON Product.SKU = Purchased.Product_SKU
10 GROUP BY
11 Product.SKU,
12 Product.Name;
13
14 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

```

1 CREATE OR REPLACE FUNCTION Update_Product_Revenue()
2 RETURNS TRIGGER AS $$ 
3 BEGIN
4     UPDATE Product_Revenue
5     SET Total_Revenue = Total_Revenue + NEW.Amount
6     WHERE SKU = NEW.Product_SKU;
7     RETURN NEW;
8 END;
9 $$ LANGUAGE plpgsql;
10
11 CREATE TRIGGER Update_Product_Revenue
12 AFTER INSERT ON Purchased
13 FOR EACH ROW
14 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

```
1 CREATE OR REPLACE FUNCTION Update_Customer_Orders()
2 RETURNS TRIGGER AS $$ 
3 BEGIN
4     UPDATE Customer_Orders
5         SET Total_Orders = Total_Orders + 1
6         WHERE Phone_number = NEW.Customer_phone_number;
7     RETURN NEW;
8 END;
9 $$ LANGUAGE plpgsql;
10
11 CREATE TRIGGER Update_Customer_Orders
12 AFTER INSERT ON Orders
13 FOR EACH ROW
14 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

```
1 CREATE OR REPLACE FUNCTION Update_Department_Employees()
2 RETURNS TRIGGER AS $$ 
3 BEGIN
4     UPDATE Department_Employees
5         SET Total_Employees = Total_Employees + 1
6         WHERE Department_name = NEW.Department_name;
7     RETURN NEW;
8 END;
9 $$ LANGUAGE plpgsql;
10
11 CREATE TRIGGER Update_Department_Employees
12 AFTER INSERT ON Employee
13 FOR EACH ROW
14 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

```

1 CREATE OR REPLACE FUNCTION Update_Department_Employees_Remove()
2 RETURNS TRIGGER AS $$ 
3 BEGIN
4     UPDATE Department_Employees
5     SET Total_Employees = Total_Employees - 1
6     WHERE Department_name = OLD.Department_name;
7     RETURN OLD;
8 END;
9 $$ LANGUAGE plpgsql;
10
11 CREATE TRIGGER Update_Department_Employees_Remove
12 AFTER DELETE ON Employee
13 FOR EACH ROW
14 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

```

1 CREATE OR REPLACE FUNCTION Update_Product_Revenue_Sold()
2 RETURNS TRIGGER AS $$ 
3 BEGIN
4     UPDATE Product_Revenue
5     SET Total_Revenue = Total_Revenue + NEW.Amount
6     WHERE SKU = NEW.Product_SKU;
7     RETURN NEW;
8 END;
9 $$ 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 CREATE OR REPLACE FUNCTION Get_Product_Revenue(Product_SK VC)
2 RETURNS NUMERIC AS $$ 
3 DECLARE
4     Total_Revenue NUMERIC;
5 BEGIN
6     SELECT SUM(Amount) INTO Total_Revenue
7     FROM Purchased
8     WHERE Purchased.Product_SKU = Product_SK;
9     RETURN Total_Revenue;
10 END;
11 $$ LANGUAGE plpgsql;
12
13 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.

The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for 'Query' (which is selected) and 'Query History'. Below the tabs is the SQL code for creating the function:

```

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

```

Below the code is a toolbar with various icons. Underneath the toolbar is a table showing the result of the function execution:

	get_product_revenue	numeric
1		1200

Figure 11.34: *Get Product Revenue Function*

11.4.2 Get Customer Orders Function

```

1 CREATE OR REPLACE FUNCTION Get_Customer_Orders(Customer_phone_number
      VARCHAR)
2 RETURNS INT AS $$ 
3 DECLARE
4   Total_Orders INT;
5 BEGIN
6   SELECT COUNT(Order_id) INTO Total_Orders
7   FROM Orders
8   WHERE Orders.Customer_phone_number = Get_Customer_Orders.
9     Customer_phone_number;
9   RETURN Total_Orders;
10 END;
11 $$ LANGUAGE plpgsql;
12
13 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.

Query History

```
1+ CREATE OR REPLACE FUNCTION Get_Customer_Orders(Customer_phone_number VARCHAR)
2 RETURNS INT AS $$ 
3 DECLARE
4     Total_Orders INT;
5 BEGIN
6     SELECT COUNT(Order_id) INTO Total_Orders
7     FROM Orders
8     WHERE Orders.Customer_phone_number = Get_Customer_Orders.Customer_phone_number;
9     RETURN Total_Orders;
10 END;
11 $$ LANGUAGE plpgsql;
12
13 SELECT Get_Customer_Orders('+96134567890');
```

Data Output Messages Notifications

get_customer_orders
integer

	get_customer_orders	lock
1	2	

Figure 11.35: *Get Customer Orders Function*

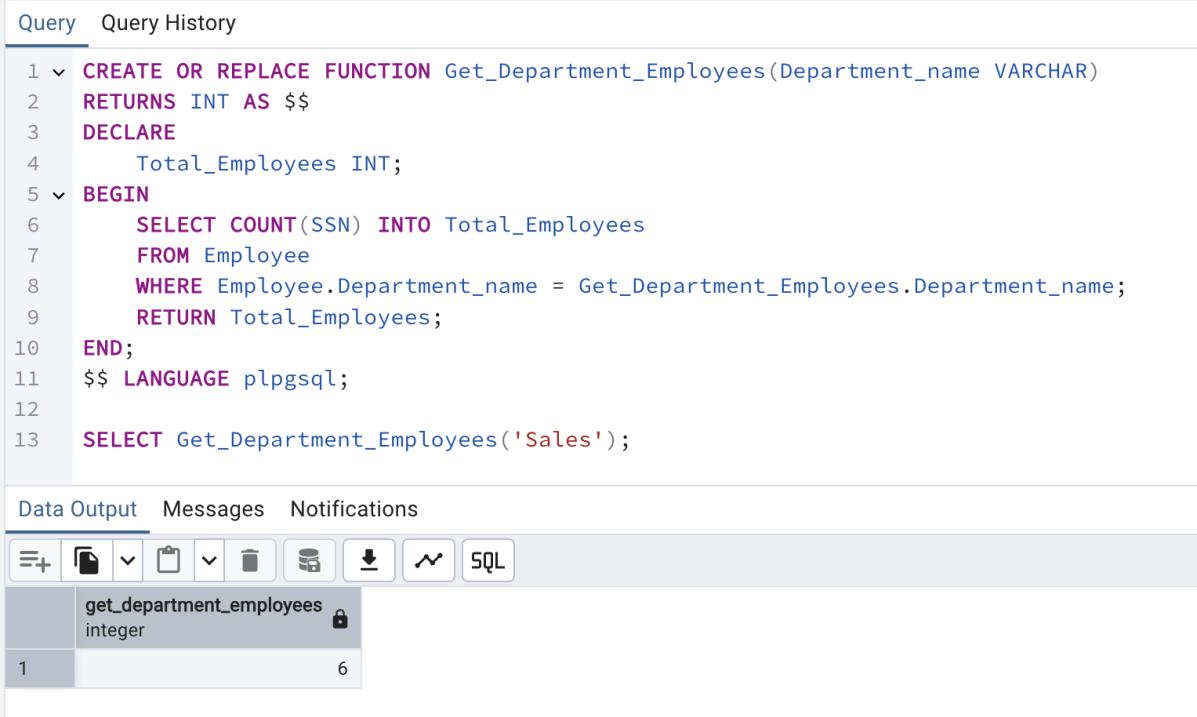
11.4.3 Get Department Employees Function

```
1 CREATE OR REPLACE FUNCTION Get_Department_Employees(Department_name VARCHAR
2 ) )
3 RETURNS INT AS $$ 
4 DECLARE
5     Total_Employees INT;
6 BEGIN
7     SELECT COUNT(SSN) INTO Total_Employees
8     FROM Employee
9     WHERE Employee.Department_name = Get_Department_Employees.
10        Department_name;
11     RETURN Total_Employees;
12 END;
13 $$ LANGUAGE plpgsql;
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
478
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1995
1996
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2095
2096
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2195
2196
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2
```

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.



The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for 'Query' (which is selected) and 'Query History'. Below the tabs is a code editor containing the following SQL code:

```

1 CREATE OR REPLACE FUNCTION Get_Department_Employees(Department_name VARCHAR)
2 RETURNS INT AS $$ 
3 DECLARE
4     Total_Employees INT;
5 BEGIN
6     SELECT COUNT(ssn) INTO Total_Employees
7     FROM Employee
8     WHERE Employee.Department_name = Get_Department_Employees.Department_name;
9     RETURN Total_Employees;
10 END;
11 $$ LANGUAGE plpgsql;
12
13 SELECT Get_Department_Employees('Sales');

```

Below the code editor is a 'Data Output' tab, followed by 'Messages' and 'Notifications' tabs. Under the 'Data Output' tab, there is a toolbar with various icons. A table is displayed with one row of data:

	get_department_employees	integer
1		6

Figure 11.36: *Get Department Employees Function*

11.5 Stored Procedures

11.5.1 Calculate Category Revenue Procedure

```

1 CREATE OR REPLACE PROCEDURE Calculate_Category_Revenue(Category_name
    VARCHAR)
2 AS $$ 
3 DECLARE
4     Total_Revenue NUMERIC;
5 BEGIN
6     SELECT SUM(Purchased.Amount) INTO Total_Revenue
7     FROM Purchased
8     JOIN Product ON Purchased.Product_SKU = Product.SKU
9     WHERE Product.Category_name = Calculate_Category_Revenue.Category_name;
10    RAISE NOTICE 'Total revenue generated by category %: %', Category_name,
11        Total_Revenue;
12 END;
13 $$ LANGUAGE plpgsql;
14
15 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.

```

Query  Query History
1 ✓ CREATE OR REPLACE PROCEDURE Calculate_Category_Revenue(Category_name VARCHAR)
2 AS $$ 
3 DECLARE
4     Total_Revenue NUMERIC;
5 BEGIN
6     SELECT SUM(Purchased.Amount) INTO Total_Revenue
7     FROM Purchased
8     JOIN Product ON Purchased.Product_SKU = Product.SKU
9     WHERE Product.Category_name = Calculate_Category_Revenue.Category_name;
10    RAISE NOTICE 'Total revenue generated by category %: %', Category_name, Total_Revenue;
11 END;
12 $$ LANGUAGE plpgsql;
13
14 CALL Calculate_Category_Revenue('Electronics');

Data Output  Messages  Notifications
NOTICE:  Total revenue generated by category Electronics: 4460
CALL

Query returned successfully in 58 msec.

```

Figure 11.37: *Calculate Category Revenue Procedure*

12 Normalization

12.1 Branch



Figure 12.1: *Branch Table Normalization*

- 1NF is satisfied in "Branch" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Branch" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Phone_number". If the primary key is dropped the FD1 cannot hold.

- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Branch", no non-prime attribute is transitively dependent on the primary key "Phone_number".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.2 Category

Name	Description	Parent_Category_Name
FD1		

Figure 12.2: *Category Table Normalization*

- 1NF is satisfied in "Category" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Category" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Name". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Category", no non-prime attribute is transitively dependent on the primary key "Name".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.3 Colors

Product SKU	Product color

Figure 12.3: *Colors Table Normalization*

- There are no FDs, so 1NF, 2NF, 3NF and BCNF are satisfied.

12.4 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
FD1											

Figure 12.4: *Coupon Table Normalization*

- 1NF is satisfied in "Coupon" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.

- 2NF is satisfied in "Coupon" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Code". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Coupon", no non-prime attribute is transitively dependent on the primary key "Code".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.5 Customer

Phone_number	Email	First_name	Last_name	Gender	Registration_date	Password_hashed	Date_of_birth	Country	State	City	Street	Building	Apartment
FD1													

Figure 12.5: *Customer Table Normalization*

- 1NF is satisfied in "Customer" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Customer" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Phone_number". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Customer", no non-prime attribute is transitively dependent on the primary key "Phone_number".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.6 Department Table

Name	Number_of_employees	Employee_SSN	Manager_start_date
FD1			

Figure 12.6: *Department Table Normalization*

- 1NF is satisfied in "Department" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Department" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Name". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Department", no non-prime attribute is transitively dependent on the primary key "Name".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.7 Department Location Table

<u>Department name</u>	<u>Location</u>
------------------------	-----------------

Figure 12.7: *Department Location Table Normalization*

- There are no FDs, so 1NF, 2NF, 3NF and BCNF are satisfied.

12.8 Dependent Table

<u>Employee SSN</u>	<u>Name</u>	<u>Gender</u>	<u>Date_of_birth</u>	<u>Relationship</u>
FD1				

Figure 12.8: *Dependent Table Normalization*

- 1NF is satisfied in "Dependent" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Dependent" since it is in 1NF, and all non-prime attributes depend fully on the composite primary key {Employee_SSN, Name}. If part of the primary key is dropped the FD1 cannot hold.
- 3NF is not satisfied since, although it is in 2NF, there is a Transitive FD in the entity "Dependent", where a non-prime attribute "Gender" is transitively dependent on the composite primary key {Employee_SSN, Name} through the non-prime attribute "Relationship".

12.8.1 D1

<u>Employee SSN</u>	<u>Name</u>	<u>Date_of_birth</u>	<u>Relationship</u>
FD1			

Figure 12.9: *D1 Table Normalization*

- 1NF is satisfied in "D1" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.

- 2NF is satisfied in "D1" since it is in 1NF, and all non-prime attributes depend fully on the primary key {Employee, Name}. If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "D1", no non-prime attribute is transitively dependent on the primary key {Employee, Name}.
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.8.2 D2

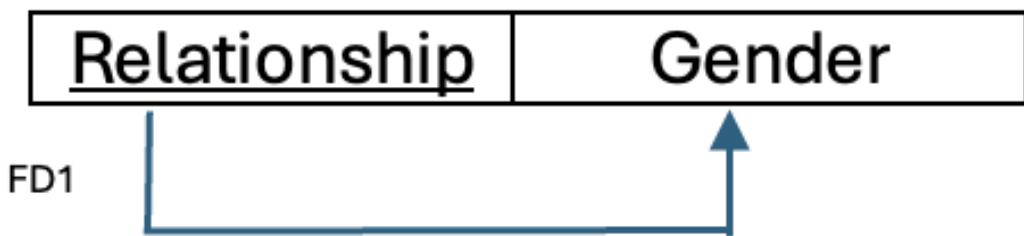


Figure 12.10: *D2 Table Normalization*

- 1NF is satisfied in "D2" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "D2" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Relationship". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "D2", no non-prime attribute is transitively dependent on the primary key "Relationship".
- BCNF is satisfied since it is in 3NF and has no FD2 $X \rightarrow A$ such that X is not a super key.

12.9 Driver Table

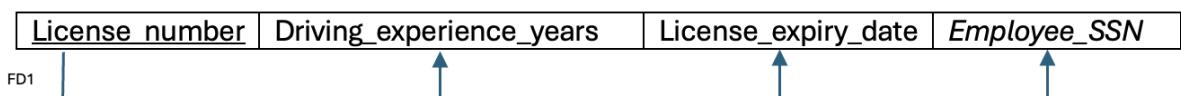


Figure 12.11: *Driver Table Normalization*

- 1NF is satisfied in "Driver" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Driver" since it is in 1NF, and all non-prime attributes depend fully on the primary key "License_number". If the primary key is dropped the FD1 cannot hold.

- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Driver", no non-prime attribute is transitively dependent on the primary key "License_number".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.10 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
FD1																		

Figure 12.12: *Employee Table Normalization*

- 1NF is satisfied in "Employee" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Employee" since it is in 1NF, and all non-prime attributes depend fully on the primary key "SSN". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Employee", no non-prime attribute is transitively dependent on the primary key "SSN".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.11 Located In Table

Product_SKU	Branch-phone-number	Quantity	Shelf_Location
FD			

Figure 12.13: *Located In Table Normalization*

- 1NF is satisfied in "Located in" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Located in" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Product_SKU". If the primary key is dropped the FD11 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Located in", no non-prime attribute is transitively dependent on the primary key "Product_SKU".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.12 Orders Table

Order_id	Notes	Payment_method	Total_amount	Is_online	Employee_SSN	Customer_phone_number	Date	Driver_License_number
FD								

Figure 12.14: *Orders Table Normalization*

- 1NF is satisfied in "Orders" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Orders" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Order_id". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD1 in the entity "Orders", no non-prime attribute is transitively dependent on the primary key "Order_id".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.13 Product Table

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

Figure 12.15: *Product Table Normalization*

- 1NF is satisfied in "Product" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Product" since it is in 1NF, and all non-prime attributes depend fully on the primary key "SKU". If the primary key is dropped the FD1 cannot hold.
- 3NF is not satisfied since, although it is in 2NF, there are non-prime attributes that are transitively dependent on the primary key "SKU" in the entity "Product".

12.13.1 P1

<u>SKU</u>	Name	Price	Description
FD			

Figure 12.16: P1 Table Normalization

- 1NF is satisfied in "P1" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "P1" since it is in 1NF, and all non-prime attributes depend fully on the primary key "SKU". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD1 in the entity "P1", no non-prime attribute is transitively dependent on the primary key "SKU".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.13.2 P2

Name	Weight	Brand	Width	Height	Length	Category_name
FD						

Figure 12.17: P2 Table Normalization

- 1NF is satisfied in "P2" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "P2" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Name". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD1 in the entity "P2", no non-prime attribute is transitively dependent on the primary key "Name".
- BCNF is satisfied since it is in 3NF and has no FD2 $X \rightarrow A$ such that X is not a super key.

12.13.3 P3

<u>Brand</u>	<u>Supplier_website</u>
FD	

Figure 12.18: *P3 Table Normalization*

- 1NF is satisfied in "P3" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "P3" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Brand". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD1 in the entity "P3", no non-prime attribute is transitively dependent on the primary key "Brand".
- BCNF is satisfied since it is in 3NF and has no FD2 $X \rightarrow A$ such that X is not a super key.

12.14 Product Image URLs Table

<u>Product SKU</u>	<u>Product Image URL</u>

Figure 12.19: *Product Image URLs Table Normalization*

- There are no FDs, so 1NF, 2NF, 3NF and BCNF are satisfied.

12.15 Purchased Table

Product_SKU	Order_id	Quantity	Amount
FD			

Figure 12.20: *Purchased Table Normalization*

- 1NF is satisfied in "Purchased" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Purchased" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Product_SKU".
- 3NF is satisfied since it is in 2NF and there is no Transitive FD1 in the entity "Purchased", no non-prime attribute is transitively dependent on the primary key "Product_SKU".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.16 Reviews Table

Product_SKU	Customer_phone_number	Review_date	Rating	Comment	Description
FD					

Figure 12.21: *Reviews Table Normalization*

- 1NF is satisfied in "Reviews" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Reviews" since it is in 1NF, and all non-prime attributes depend fully on the primary key {Product_SKU, Customer.phone_number}. If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Reviews", no non-prime attribute is transitively dependent on the primary key {Product_SKU, Customer.phone_number}.
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.17 Review Image URLs Table

<u>Product_SKU</u>	<u>Customer_phone_number</u>	<u>Product_Image_URL</u>

Figure 12.22: *Review Image URLs Table Normalization*

- 1NF is satisfied in "Review Image URLs" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Review Image URLs" since it is in 1NF, and all non-prime attributes depend fully on the primary key {Product_SKU, Product_Image_URL}. If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Review Image URLs", no non-prime attribute is transitively dependent on the primary key {Product_SKU, Product_Image_URL}.
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.18 Support Ticket Table

<u>Ticket_id</u>	<u>Description</u>	<u>Subject</u>	<u>Status</u>	<u>Priority</u>	<u>Employee_SSN</u>	<u>Customer_phone_number</u>

Figure 12.23: *Support Ticket Table Normalization*

- 1NF is satisfied in "Support Ticket" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Support Ticket" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Ticket_id". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Support Ticket", no non-prime attribute is transitively dependent on the primary key "Ticket_id".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.19 Supplier Table

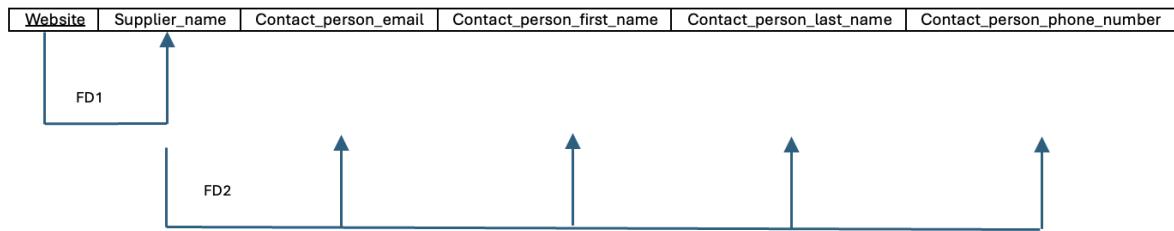


Figure 12.24: *Supplier Table Normalization*

- 1NF is satisfied in "Supplier" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Supplier" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Website". If the primary key is dropped the FD1 cannot hold.
- 3NF is not satisfied since, although it is in 2NF, there are non-prime attributes that are transitively dependent on the primary key "Website" in the entity "Supplier".

12.19.1 S1

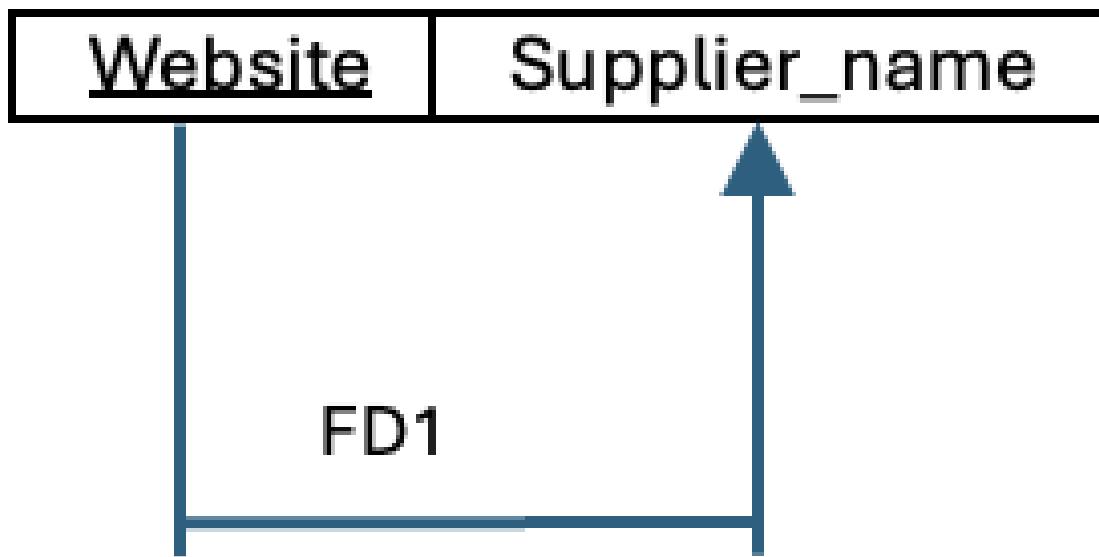


Figure 12.25: *S1 Table Normalization*

- 1NF is satisfied in "S1" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.

- 2NF is satisfied in "S1" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Website". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "S1", no non-prime attribute is transitively dependent on the primary key "Website".
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.19.2 S2

Supplier_name	Contact_person_email	Contact_person_first_name	Contact_person_last_name	Contact_person_phone_number
FD2				

Figure 12.26: S2 Table Normalization

- 1NF is satisfied in "S2" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "S2" since it is in 1NF, and all non-prime attributes depend fully on the primary key "Supplier_name". If the primary key is dropped the FD1 cannot hold.
- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "S2", no non-prime attribute is transitively dependent on the primary key "Supplier_name".
- BCNF is satisfied since it is in 3NF and has no FD2 $X \rightarrow A$ such that X is not a super key.

12.20 Wishlist Table

SKU	Customer_phone_number	Total_amount

Figure 12.27: Wishlist Table Normalization

- 1NF is satisfied in "Wishlist" since all values are single atomic (or indivisible) values, and there are no multivalued or composite attributes in this entity.
- 2NF is satisfied in "Wishlist" since it is in 1NF, and all non-prime attributes depend fully on the primary key {SKU, Customer_phone_number}. If the primary key is dropped the FD1 cannot hold.

- 3NF is satisfied since it is in 2NF and there is no Transitive FD in the entity "Wishlist", no non-prime attribute is transitively dependent on the primary key {SKU, Customer_phone_number}.
- BCNF is satisfied since it is in 3NF and has no FD1 $X \rightarrow A$ such that X is not a super key.

12.21 Working Hours Table

<u>Branch phone number</u>	<u>Day</u>	<u>Opening hour</u>	<u>Closing hour</u>
----------------------------	------------	---------------------	---------------------

Figure 12.28: *Working Table Normalization*

- There are no FDs, so 1NF, 2NF, 3NF and BCNF are satisfied.

13 Conclusion

The completion of Phase 4 of the project signifies a major milestone in the learning journey of database systems. By focusing on normalization, we have gained a deeper understanding of the principles that ensure database efficiency and consistency. Through the process of analyzing and normalizing database schemas to meet various normal forms (1NF, 2NF, 3NF, and BCNF), we have developed the ability to eliminate redundancy, prevent anomalies, and ensure data integrity. This phase not only reinforces theoretical concepts but also equips students with the practical skills required to design normalized, well-structured, and scalable database systems for diverse real-world applications.