

Foundations of Databases A.Y. 2023-2024

Homework 3 – Physical Design

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: December 15, 2023

Team acronym	ECOMMESTA	
Last Name	First Name	Student Number
CAKMAKCI	UMUT BERK	2071408
YANOGLU	MELTEM	2071545
TORRES-PARDO LÓPEZ	ISABEL	2099920
OZGEN	ALARA SELEN	2071410
VARELA MARTÍN	ALEJANDRO	2100062
MALOCHE	DESALEGN MATHEWOS	2090063

Variations to the Relational Schema

The Figure 1 shows the relational schema after some changes made due to the feedback received in the correction of the second homework.

- We create a new relationship between Receipt and Employee, called **Emitted-By**, a one-to-many relationship created to praise the importance of Employee.
- We changed the cardinality of the relation between *Product* and *Supplier*, from a one-to-many to a many-to-many.
- We changed the relationship **Supplies-to**, a binary relationship between *Product* and *Supplier*, to **Supplies**, a ternary relationship between *Supplier*, *Product* and *Store*, with the attribute Quantity, that indicated the amount of product supplied by a supplier to a specific store.
- We deleted the attribute *orderquantity* of the relationship Stores, because now this information in the *Supplies* relationship's attribute, **quantity**.

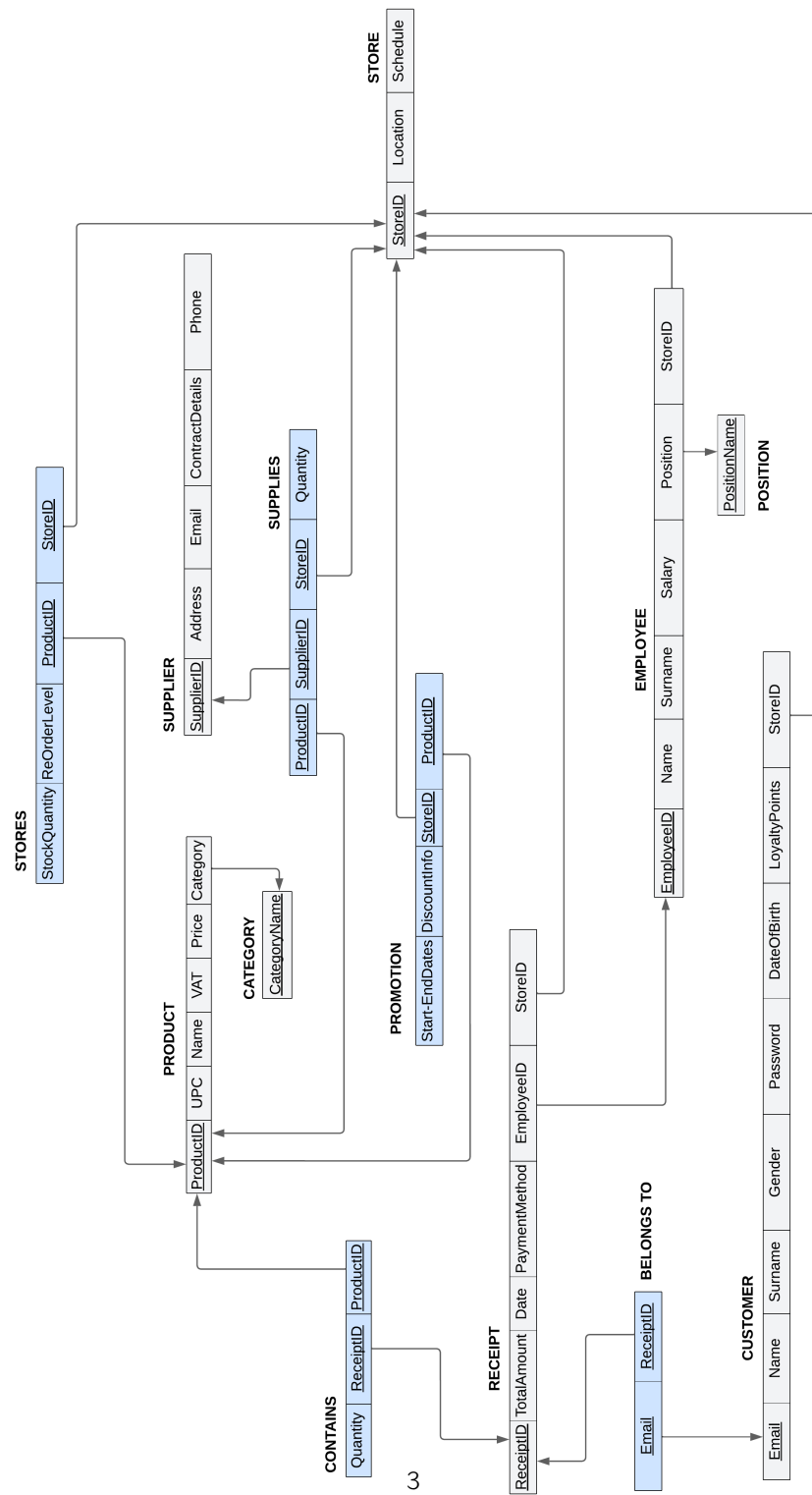


Figure 1: Transformed Relational Schema

Physical Schema

In the following part, there will be the SQL instructions needed to create the database explained above.

```
-- Creation of the Mercadone database schema

-- #####
-- ## Database creation                                ##
-- #####

-- Delete pre-existent database instance (if any)
DROP DATABASE IF EXISTS Mercadone;

-- Create the database
CREATE DATABASE Mercadone ENCODING 'UTF-8';

-- #####
-- ## Database connection                                ##
-- #####

-- Connect to the database
\connect Mercadone

-- #####
-- ## Domains creation                                    ##
-- #####

-- Create new domains
CREATE DOMAIN passwd AS VARCHAR(254)
  CONSTRAINT properpassword CHECK (((VALUE)::text ~* '[A-Za-z0-9._%!]{8,}'))::text));

CREATE DOMAIN emailaddress AS VARCHAR(254)
  CONSTRAINT properemail CHECK (((VALUE)::text ~* '^[A-Za-z0-9._%]+@[A-Za-z0-9.]+\.[A-Za-z]+$'))::text));

-- #####
-- ## Types creation                                    ##
-- #####

-- Create new types
CREATE TYPE timerange AS RANGE (subtype = time);

CREATE TYPE gender AS ENUM (
    'Male',
    'Female',
    'Other',
    'I prefer not to say'
);

CREATE TYPE paymentmethod AS ENUM (
```

```

        'CASH',
        'CARD'
    );

-- #####
-- ## Tables creation ##
-- #####

-- This table represents a category
CREATE TABLE Category (
    categoryname CHARACTER VARYING (250) PRIMARY KEY
);

COMMENT ON TABLE Category IS 'Categories to which products belong.';
COMMENT ON COLUMN Category.categoryname IS 'Unique identifier for the category.';

-- This table represents a product
CREATE TABLE Product (
    productid SERIAL PRIMARY KEY,
    upc BIGINT NOT NULL,
    name CHARACTER VARYING (250) NOT NULL,
    vat REAL NOT NULL,
    price DECIMAL(19, 4) NOT NULL,
    category CHARACTER VARYING (250) NOT NULL,
    FOREIGN KEY ( category ) REFERENCES Category (categoryname)
);

COMMENT ON TABLE Product IS 'Represents a product.';
COMMENT ON COLUMN Product.productid IS 'Unique identifier for the product.';
COMMENT ON COLUMN Product.upc IS 'Universal Product Code.';
COMMENT ON COLUMN Product.name IS 'Name of the product.';
COMMENT ON COLUMN Product.vat IS 'Value Added Tax.';
COMMENT ON COLUMN Product.price IS 'Price of the product.';
COMMENT ON COLUMN Product.category IS 'Unique identifier for the category of the product.';

-- This table represents a store
CREATE TABLE Store (
    storeid SERIAL PRIMARY KEY,
    location CHARACTER VARYING (250) NOT NULL,
    schedule timerange[7] NOT NULL
);

COMMENT ON TABLE Store IS 'Represents a store.';
COMMENT ON COLUMN Store.storeid IS 'Unique identifier for the store.';
COMMENT ON COLUMN Store.location IS 'Address of the store.';
COMMENT ON COLUMN Store.schedule IS 'Timetable of the store.';

-- This table represents the quantity of a specific product stored in a store and its re-order level
CREATE TABLE Stores (
    productid SERIAL,

```

```

        storeid SERIAL,
        stockquantity INTEGER NOT NULL,
        reorderlevel INTEGER NOT NULL,
        FOREIGN KEY ( productid ) REFERENCES Product (productid),
        FOREIGN KEY ( storeid ) REFERENCES Store (storeid),
        PRIMARY KEY ( productid, storeid )
    );

COMMENT ON TABLE Stores IS 'Represents the quantity of a specific product stored in a store
and its re-order level.';
COMMENT ON COLUMN Stores.productid IS 'Unique identifier for the product.';
COMMENT ON COLUMN Stores.storeid IS 'Unique identifier for the store.';
COMMENT ON COLUMN Stores.stockquantity IS 'Quantity available in the store.';
COMMENT ON COLUMN Stores.reorderlevel IS 'Minimum quantity that must be in the store.';

-- This table represents the discounts for the products in a store
CREATE TABLE Promotion (
    productid SERIAL,
    storeid SERIAL,
    startenddates DATERANGE NOT NULL,
    discountinfo CHARACTER VARYING (250) NOT NULL,
    FOREIGN KEY ( productid ) REFERENCES Product (productid),
    FOREIGN KEY ( storeid ) REFERENCES Store (storeid),
    PRIMARY KEY ( productid, storeid )
);

COMMENT ON TABLE Promotion IS 'Represents a promotion.';
COMMENT ON COLUMN Promotion.productid IS 'Unique identifier for the product.';
COMMENT ON COLUMN Promotion.storeid IS 'Unique identifier for the store.';
COMMENT ON COLUMN Promotion.startenddates IS 'Initial and final date of a promotion.';
COMMENT ON COLUMN Promotion.discountinfo IS 'Information or description of the promotion.';

-- This table represents a supplier
CREATE TABLE Supplier (
    supplierid SERIAL PRIMARY KEY,
    address CHARACTER VARYING (250) NOT NULL,
    email EMAILADDRESS NOT NULL,
    contractdetails CHARACTER VARYING (250) NOT NULL,
    phone CHARACTER VARYING (25) NOT NULL
);

COMMENT ON TABLE Supplier IS 'Represents a supplier.';
COMMENT ON COLUMN Supplier.supplierid IS 'Unique identifier for the supplier.';
COMMENT ON COLUMN Supplier.address IS 'Address of the supplier.';
COMMENT ON COLUMN Supplier.email IS 'Email of the supplier.';
COMMENT ON COLUMN Supplier.contractdetails IS 'Details and description of the contract between store
and supplier.';
COMMENT ON COLUMN Supplier.phone IS 'Phone number of the supplier.';

-- This table represents the quantity of a specific product supply by a supplier to a store
CREATE TABLE Supplies (

```

```

        productid SERIAL,
        storeid SERIAL,
        supplierid SERIAL,
        quantity INTEGER NOT NULL,
        FOREIGN KEY ( productid ) REFERENCES Product (productid),
        FOREIGN KEY ( storeid ) REFERENCES Store (storeid),
        FOREIGN KEY ( supplierid ) REFERENCES Supplier (supplierid),
        PRIMARY KEY ( productid, storeid, supplierid )
    );

COMMENT ON TABLE Supplies IS 'Represents the quantity of a specific product supply
by a supplier to a store.';
COMMENT ON COLUMN Supplies.productid IS 'Unique identifier for the product.';
COMMENT ON COLUMN Supplies.storeid IS 'Unique identifier for the store.';
COMMENT ON COLUMN Supplies.supplierid IS 'Unique identifier for the supplier.';
COMMENT ON COLUMN Supplies.quantity IS 'Quantity ordered.';

-- This table represents a position
CREATE TABLE Position (
    positionname CHARACTER VARYING (250) PRIMARY KEY
);

COMMENT ON TABLE Position IS 'A title of an employee.';
COMMENT ON COLUMN Position.positionname IS 'Unique identifier for the position.';

-- This table represents an employee
CREATE TABLE Employee (
    employeeid SERIAL PRIMARY KEY,
    name CHARACTER VARYING (250) NOT NULL,
    surname CHARACTER VARYING (250) NOT NULL,
    salary DECIMAL(19, 4) NOT NULL,
    position CHARACTER VARYING (250) NOT NULL,
    storeid SERIAL NOT NULL,
    FOREIGN KEY ( position ) REFERENCES Position (positionname),
    FOREIGN KEY ( storeid ) REFERENCES Store (storeid)
);

COMMENT ON TABLE Employee IS 'Represents an employee.';
COMMENT ON COLUMN Employee.employeeid IS 'Unique identifier for the employee.';
COMMENT ON COLUMN Employee.name IS 'Name of the employee.';
COMMENT ON COLUMN Employee.surname IS 'Surname of the employee.';
COMMENT ON COLUMN Employee.salary IS 'Salary of the employee.';
COMMENT ON COLUMN Employee.position IS 'Unique identifier for the position of the employee.';
COMMENT ON COLUMN Employee.storeid IS 'Unique identifier for the store where the employee works.';

-- This table represents a receipt
CREATE TABLE Receipt (
    receiptid SERIAL PRIMARY KEY,
    totalamount DECIMAL(19, 4) NOT NULL,
    date TIMESTAMP NOT NULL,
    paymentmethod PAYMENTMETHOD NOT NULL,

```

```

        storeid SERIAL NOT NULL,
        employeeid SERIAL NOT NULL,
        FOREIGN KEY ( storeid ) REFERENCES Store (storeid),
        FOREIGN KEY ( employeeid ) REFERENCES Employee (employeeid)
    );

COMMENT ON TABLE Receipt IS 'Represents a receipt.';
COMMENT ON COLUMN Receipt.receiptid IS 'Unique identifier for the receipt.';
COMMENT ON COLUMN Receipt.totalamount IS 'Total price of the purchase.';
COMMENT ON COLUMN Receipt.date IS 'Date of the purchase.';
COMMENT ON COLUMN Receipt.paymentmethod IS 'Type of payment used.';
COMMENT ON COLUMN Receipt.storeid IS 'Unique identifier for the store where the purchase was made.';
COMMENT ON COLUMN Receipt.employeeid IS 'Unique identifier for the employee who attended during
the purchase.';

-- This table represents the quantity of a specific product contained in a receipt
CREATE TABLE Contains (
    receiptid SERIAL,
    productid SERIAL,
    quantity INTEGER NOT NULL,
    FOREIGN KEY ( receiptid ) REFERENCES Receipt (receiptid),
    FOREIGN KEY ( productid ) REFERENCES Product (productid),
    PRIMARY KEY ( receiptid, productid )
);

COMMENT ON TABLE Contains IS 'Represents the quantity of a specific product contained in a receipt.';
COMMENT ON COLUMN Contains.receiptid IS 'Unique identifier for the receipt.';
COMMENT ON COLUMN Contains.productid IS 'Unique identifier for the product.';
COMMENT ON COLUMN Contains.quantity IS 'Quantity of product bought.';

-- This table represents a customer
CREATE TABLE Customer (
    email EMAILADDRESS PRIMARY KEY,
    name CHARACTER VARYING (250) NOT NULL,
    surname CHARACTER VARYING (250) NOT NULL,
    gender GENDER NOT NULL,
    password CHARACTER VARYING (250) NOT NULL,
    dateofbirth DATE NOT NULL,
    loyaltypoints INTEGER NOT NULL,
    storeid SERIAL,
    FOREIGN KEY ( storeid ) REFERENCES Store (storeid)
);

COMMENT ON TABLE Customer IS 'Represents a customer.';
COMMENT ON COLUMN Customer.email IS 'Unique identifier for the customer.';
COMMENT ON COLUMN Customer.name IS 'Name of the customer.';
COMMENT ON COLUMN Customer.surname IS 'Surname of the customer.';
COMMENT ON COLUMN Customer.gender IS 'Gender of the customer.';
COMMENT ON COLUMN Customer.password IS 'Password of the customer.';
COMMENT ON COLUMN Customer.dateofbirth IS 'Date of Birth of the customer.';
COMMENT ON COLUMN Customer.loyaltypoints IS 'Points obtained by the customer.';
COMMENT ON COLUMN Customer.storeid IS 'Unique identifier for the store preferred by the customer.';

```



```

-- This table represents which receipt belongs to which customer (if customer used the loyalty program while
CREATE TABLE BelongsTo (
    receiptid SERIAL PRIMARY KEY,
    email EMAILADDRESS,
    FOREIGN KEY ( receiptid ) REFERENCES Receipt (receiptid),
    FOREIGN KEY ( email ) REFERENCES Customer (email)
);

COMMENT ON TABLE BelongsTo IS 'Associates the customer with the receipt
(if customer used the loyalty program while purchasing).';
COMMENT ON COLUMN BelongsTo.receiptid IS 'Unique identifier for the receipt.';
COMMENT ON COLUMN BelongsTo.email IS 'Unique identifier for the product.';

```

Populate the Database: Example

In the following, there are some examples of SQL instructions to insert data in the tables.

```

--populating the database
INSERT INTO Category (categoryname) VALUES
    ('Fresh Produce'),
    ('Bakery and Pastries'),
    ('Dairy and Eggs'),
    ('Meat and Seafood'),
    ('Frozen Foods'),
    ('Grocery and Staples'),
    ('Beverages'),
    ('Snacks'),
    ('Cereal and Breakfast Foods'),
    ('Condiments and Sauces'),
    ('Household and Cleaning Supplies'),
    ('Health and Beauty'),
    ('Pet Supplies'),
    ('Baby Care'),
    ('Electronics and Appliances');

INSERT INTO Product (productid, upc, name, vat, price, category) VALUES
    (01001, 123456789012, 'FreshHarvest Organic Golden Apple', 0.04, 2.99, 'Fresh Produce'),
    (03001, 345678901234, 'Whoole Milk', 0.12, 4.49, 'Dairy and Eggs'),
    (04001, 456789012345, 'DeliMaster Smoked Chicken Breast', 0.22, 8.89, 'Meat and Seafood'),
    ,
    (05001, 567890123456, 'FreezeMaster Supreme Frozen Pizza', 0.22, 5.49, 'Frozen Foods'),
    (06001, 678901234567, 'NatureHarbor Organic Canned Beans', 0.10, 2.29, 'Grocery and
    Staples'),
    (07001, 789012345678, 'Pepsi Cola', 0.22, 2.29, 'Beverages'),
    (08001, 890123456789, 'Lays Potato Chips', 0.10, 3.19, 'Snacks'),
    (09001, 901234567890, 'Kelligs Oatmeal', 0.04, 4.20, 'Cereal and Breakfast Foods'),
    (10001, 112233445566, 'Heinz Ketchup', 0.10, 2.50, 'Condiments and Sauces'),
    (11001, 223344556677, 'Lily Paper Towels', 0.22, 3.40, 'Household and Cleaning Supplies')
    ,

```

(12001, 334455667788, 'Loreal Paris Shampoo', 0.22, 5.70, 'Health and Beauty'),
 (13001, 445566778899, 'Whiskas Premium Dog Food', 0.10, 4.79, 'Pet Supplies'),
 (14001, 556677889900, 'Garnier RevitalizingDiapers', 0.04, 8.49, 'Baby Care'),
 (15001, 667788990011, 'Varta AA Batteries', 0.22, 6.99, 'Electronics and Appliances'),
 (01002, 778899001122, 'EcoFood Orange', 0.04, 3.0, 'Fresh Produce'),
 (02001, 889900112233, 'SweetBites Butter Croissant', 0.10, 2.0, 'Bakery and Pastries'),
 (03002, 990011223344, 'HappyCow Cheddar Cheese', 0.12, 6.49, 'Dairy and Eggs'),
 (04002, 101122334455, 'DeliMaster Smoked Salmon', 0.22, 12.0, 'Meat and Seafood'),
 (01003, 121212121212, 'GreenGrove Broccoli Bundle', 0.04, 3.39, 'Fresh Produce'),
 (02002, 131313131313, 'Frenchs Bagel', 0.10, 2.0, 'Bakery and Pastries'),
 (03003, 141414141414, 'Sutas Turkish Yogurt', 0.12, 2.49, 'Dairy and Eggs'),
 (04003, 151515151515, 'DeliMaster Shrimp', 0.22, 15.69, 'Meat and Seafood'),
 (05002, 161616161616, 'Magnum Ice Cream', 0.22, 6.0, 'Frozen Foods'),
 (06002, 171717171717, 'Barilla Pasta', 0.10, 1.70, 'Grocery and Staples'),
 (07002, 181818181818, 'Orange Juice', 0.22, 3.90, 'Beverages'),
 (10002, 212121212121, 'Heinz Mustard', 0.10, 2.40, 'Condiments and Sauces'),
 (13002, 242424242424, 'Whiskas Premium Cat Food', 0.10, 5.60, 'Pet Supplies'),
 (10003, 363636363636, 'Mexican Salsa', 0.10, 2.0, 'Condiments and Sauces'),
 (12002, 383838383838, 'Nivea Deodorant', 0.22, 5.55, 'Health and Beauty'),
 (13003, 393939393939, 'FurryCare Clumping Cat Litter', 0.10, 3.15, 'Pet Supplies'),
 (14002, 404040404040, 'Baby Wipes', 0.04, 7.65, 'Baby Care'),
 (02003, 434343434343, 'Sunrise Baguette', 0.10, 2.75, 'Bakery and Pastries'),
 (03004, 444444444444, 'Happy Cow Yogurt', 0.12, 2.29, 'Dairy and Eggs'),
 (04004, 454545454545, 'Ocean Catch Shrimp', 0.22, 15.0, 'Meat and Seafood'),
 (05003, 464646464646, 'Arctic Bliss Ice Cream', 0.22, 6.0, 'Frozen Foods'),
 (06003, 474747474747, 'Mountain Harvest Pasta', 0.10, 1.79, 'Grocery and Staples'),
 (07003, 484848484848, 'Sunset Orange Juice', 0.22, 3.35, 'Beverages'),
 (08002, 494949494949, 'SnackMaster Pretzels', 0.10, 2.25, 'Snacks'),
 (09002, 505050505050, 'Natures Granola Bars', 0.04, 4.75, 'Cereal and Breakfast Foods'),
 (10004, 515151515151, 'Spicy Delight Mustard', 0.10, 2.80, 'Condiments and Sauces'),
 (11002, 525252525252, 'Clean n Fresh Dish Soap', 0.22, 4.40, 'Household and Cleaning
Supplies'),
 (12003, 535353535353, 'Silky Smooth Soap', 0.22, 3.60, 'Health and Beauty'),
 (13004, 545454545454, 'Pawsome Cat Food', 0.10, 5.0, 'Pet Supplies'),
 (14003, 555555555555, 'Little Ones Baby Formula', 0.04, 10.79, 'Baby Care'),
 (15002, 565656565656, 'TechMaster Toaster', 0.22, 25.0, 'Electronics and Appliances'),
 (01004, 575757575757, 'Sweet Vineyard Grapes', 0.04, 3.30, 'Fresh Produce'),
 (02004, 585858585858, 'Butterfly Bakery Cinnamon Roll', 0.10, 2.50, 'Bakery and Pastries'
),
 (03005, 595959595959, 'Farm Fresh Butter', 0.10, 4.19, 'Dairy and Eggs'),
 (04005, 606060606060, 'Harvest Turkey', 0.22, 10.79, 'Meat and Seafood'),
 (05004, 616161616161, 'Arctic Frost Frozen Vegetables', 0.22, 3.59, 'Frozen Foods'),
 (06004, 626262626262, 'Sunrise Cereal', 0.10, 4.0, 'Grocery and Staples'),
 (07004, 636363636363, 'Sunny Lemonade', 0.22, 2.49, 'Beverages'),
 (08003, 646464646464, 'Trail Mix Treats', 0.10, 3.0, 'Snacks'),
 (09003, 656565656565, 'Morning Delight Pancake Mix', 0.04, 5.39, 'Cereal and Breakfast
Foods'),
 (10005, 666666666666, 'FireBlast Hot Sauce', 0.10, 2.54, 'Condiments and Sauces'),
 (11003, 676767676767, 'Clean Sweep Broom', 0.22, 8.79, 'Household and Cleaning Supplies')
 ,
 (12004, 686868686868, 'Fresh Breeze Deodorant', 0.22, 5.49, 'Health and Beauty'),
 (13005, 696969696969, 'Happy Pup Dog Toy', 0.10, 3.0, 'Pet Supplies'),
 (14004, 707070707070, 'SoftTouch Baby Wipes', 0.04, 7.49, 'Baby Care'),
 (15003, 717171717171, 'Blitz Blender', 0.22, 39.99, 'Electronics and Appliances'),

```

(15004, 129956787712, 'TechCo Smartphone X1', 0.22, 599.99, 'Electronics and Appliances')
,
(15005, 238697890123, 'ElectroTech Laptop ProBook', 0.22, 899.99, 'Electronics and
Appliances'),
(12005, 001122334455, 'Marvis Whitening Toothpaste', 0.10, 1.79, 'Health and Beauty');

```

```

INSERT INTO Store (storeid, location, schedule) VALUES

```

```

(3901, '123 Main Street', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3902, '456 Oak Avenue', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3903, '789 Maple Drive', ARRAY[
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[11:00:00, 18:00:00]':timerange,
    '[11:00:00, 16:30:00]':timerange
]),
(3904, '101 Pine Lane', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3905, '202 Cedar Street', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),

```

```

(3906, '303 Elm Road', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3907, '404 Birch Boulevard', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3908, '505 Redwood Lane', ARRAY[
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[09:00:00, 19:00:00]':timerange,
    '[11:00:00, 18:00:00]':timerange,
    '[11:00:00, 16:30:00]':timerange
]),
(3909, '606 Walnut Street', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]),
(3910, '707 Spruce Avenue', ARRAY[
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[09:00:00, 21:00:00]':timerange,
    '[11:00:00, 20:00:00]':timerange,
    '[11:00:00, 18:30:00]':timerange
]);

```

```

INSERT INTO Stores (productid, storeid, stockquantity, reorderlevel) VALUES
(01001, 3901, 10, 20),
(03001, 3902, 15, 30),
(04001, 3903, 200, 100),
(05001, 3904, 120, 25),
(06001, 3905, 18, 35),

```

```
(07001, 3906, 90, 15),
(08001, 3907, 13, 30),
(09001, 3908, 160, 20),
(10001, 3909, 110, 25),
(11001, 3910, 70, 10),
(12001, 3901, 50, 8),
(10005, 3902, 80, 12),
(11003, 3903, 6, 10),
(12004, 3904, 40, 8),
(13005, 3905, 30, 5),
(14004, 3906, 25, 4),
(15003, 3907, 20, 3),
(15004, 3908, 15, 2);
```

```
INSERT INTO Promotion (productid, storeid, startenddates, discountinfo) VALUES
(15003, 3909, '[2023-12-01, 2024-02-28]', '20% off on Blitz Blender'),
(03001, 3901, '[2024-03-10, 2024-03-31]', '25% off on Whooole Milk'),
(04001, 3902, '[2024-01-15, 2024-02-15]', '15% off on DeliMaster Smoked Chicken Breast'),
(09001, 3903, '[2024-04-10, 2024-04-30]', '30% off on Kelligs Oatmeal'),
(06001, 3904, '[2023-12-12, 2024-01-28]', '20% off on NatureHarbor Organic Canned Beans'),
,
(07001, 3905, '[2024-03-01, 2024-03-31]', '25% off on Pepsi Cola'),
(08001, 3906, '[2023-11-05, 2023-12-28]', '12% off on Lays Potato Chips'),
(10001, 3907, '[2024-01-05, 2024-02-15]', '18% off on Heinz Ketchup'),
(11001, 3908, '[2024-04-01, 2024-04-30]', '30% off on Lily Paper Towels'),
(12001, 3909, '[2023-12-10, 2024-03-15]', '15% off on Loreal Paris Shampoo'),
(13005, 3910, '[2024-03-15, 2024-04-15]', '25% off on Happy Pup Dog Toy'),
(11003, 3901, '[2024-01-10, 2024-02-10]', '10% off on Clean Sweep Broom'),
(14004, 3902, '[2024-04-01, 2024-04-15]', '18% off on SoftTouch Baby Wipes'),
(15004, 3903, '[2023-10-01, 2023-12-31]', '20% off on TechCo Smartphone X1'),
(12004, 3904, '[2024-01-01, 2024-01-31]', '10% off on Fresh Breeze Deodorant'),
(10005, 3905, '[2024-02-20, 2024-03-20]', '15% off on FireBlast Hot Sauce'),
(04001, 3906, '[2023-12-10, 2023-12-17]', '30% off on DeliMaster Smoked Chicken Breast'),
(15003, 3907, '[2023-11-05, 2023-12-28]', '12% off on Blitz Blender'),
(03001, 3908, '[2023-11-15, 2024-02-15]', '15% off on Whooole Milk'),
(08001, 3909, '[2023-12-01, 2023-12-31]', '25% off on Lays Potato Chips'),
(11001, 3910, '[2023-12-05, 2024-02-28]', '12% off on Lily Paper Towels'),
(09001, 3901, '[2024-04-01, 2024-04-30]', '30% off on Kelligs Oatmeal');
```

```
INSERT INTO Supplier (supplierid, address, email, contractdetails, phone) VALUES
(1110101, '123 Main Street, Milan, Italy', 'italianfood@example.com', 'Supply of Italian food products', '+39 02 1234567'),
(3330101, '456 Global Avenue, Worldwide', 'globalfoods@example.com', 'International food distribution', '+1 555 9876543'),
(1110102, '789 Supplier Lane, Rome, Italy', 'romefoods@example.com', 'Local specialties and produce', '+39 06 8765432'),
(2220101, '101 International Street, Paris, France', 'frenchdelights@example.com', 'French cuisine products', '+33 1 23456789'),
(4440101, '202 World Market Road, Tokyo, Japan', 'tokyogroceries@example.com', 'Japanese food supplies', '+81 3 98765432'),
(2220102, '303 International Boulevard, Madrid, Spain', 'spanishfoods@example.com', 'Authentic Spanish products', '+34 91 8765432'),
```

```
(2220103, '404 Supplier Street, Berlin, Germany', 'germanmarket@example.com', 'German
delicacies and goods', '+49 30 98765432'),
(2220104, '505 Supplier Square, London, UK', 'britishgrocery@example.com', 'British
grocery supplies', '+44 20 12345678'),
(4440102, '606 Exotic Street, Bangkok, Thailand', 'thaieexports@example.com', 'Thai food
exports', '+66 2 34567890'),
(4440103, '707 Supplier Lane, Mumbai, India', 'indianimports@example.com', 'Indian food
imports', '+91 22 23456789'),
(3330102, '808 Tropical Street, Rio de Janeiro, Brazil', 'brazilianfoods@example.com', '
Authentic Brazilian products', '+55 21 987654321'),
(4440104, '909 Exotic Lane, Seoul, South Korea', 'koreangroceries@example.com', 'Korean
food supplies', '+82 2 12345678'),
(4440105, '101 Supplier Plaza, Beijing, China', 'chinesemarket@example.com', 'Chinese
grocery distribution', '+86 10 87654321'),
(4440106, '202 Spices Street, New Delhi, India', 'indianspices@example.com', 'Indian
spices and herbs', '+91 11 23456789'),
(3330103, '303 Tech Street, San Francisco, USA', 'techgadgets@example.com', 'Electronics
and gadgets supply', '+1 415 1234567'),
(4440107, '404 Appliances Avenue, Seoul, South Korea', 'koreanappliances@example.com', '
Korean home appliances distribution', '+82 2 98765432'),
(1110103, 'Via Patriarcato, 44, Padova, Italia', 'padovagoodies@gmail.com', 'Fresh fruits
and vegetables supplier', '+39 111 1234567'),
(2220105, 'Avinguda de la Riera de Cassoles, 26, Gracia, Barcelona, Spain', '
bigbarcelonaprod@gmail.com', 'Electronic household appliances supply', '+34 123 456
789'),
(2220106, 'Beysehir Yolu Uzeri, 47, Meram, Konya, Turkey', 'torkusuturunleri@gmail.com',
'Turkish special products supplier', '+90 342 489 1236');
```

```
INSERT INTO Supplies(productid, storeid, supplierid, quantity) VALUES
(01001, 3901, 1110103, 250),
(02001, 3901, 2220102, 200),
(03001, 3902, 1110102, 300),
(04001, 3902, 4440105, 150),
(05001, 3903, 1110101, 400);
```

```
INSERT INTO Position (positionname) VALUES
('Manager'),
('Cashier'),
('Stock Clerk'),
('Marketing Coordinator'),
('Assistant Manager'),
('Sales Associate'),
('Customer Service Representative');
```

```
INSERT INTO Employee (employeeid, name, surname, salary, position, storeid) VALUES
(1001, 'John', 'Walter', 60000, 'Manager', 3901),
(1002, 'Paula', 'Smith', 30000, 'Cashier', 3902),
(1003, 'Bob', 'Johnson', 25000, 'Stock Clerk', 3903),
(1004, 'Emily', 'Davis', 70000, 'Marketing Coordinator', 3901),
(1005, 'Michael', 'Brown', 45000, 'Assistant Manager', 3901),
```

(1006, 'Sara', 'Wilson', 55000, 'Sales Associate', 3903),
 (1007, 'David', 'Lee', 35000, 'Customer Service Representative', 3904),
 (1008, 'Alice', 'Taylor', 32000, 'Cashier', 3905),
 (1009, 'Chris', 'Harris', 60000, 'Manager', 3902),
 (1010, 'Emma', 'Evans', 40000, 'Assistant Manager', 3902),
 (1011, 'Jack', 'Roberts', 50000, 'Sales Associate', 3902),
 (1012, 'Sophie', 'Miller', 75000, 'Cashier', 3903),
 (1013, 'Matthew', 'Clark', 42000, 'Marketing Coordinator', 3904),
 (1014, 'Olivia', 'Johnson', 48000, 'Stock Clerk', 3905),
 (1015, 'Daniel', 'White', 38000, 'Sales Associate', 3906),
 (1016, 'Jessica', 'Moore', 32000, 'Cashier', 3901),
 (1017, 'David', 'Wilson', 33000, 'Cashier', 3904),
 (1018, 'Mia', 'Martin', 55000, 'Sales Associate', 3903),
 (1019, 'Ethan', 'Garcia', 35000, 'Customer Service Representative', 3904),
 (1020, 'Ava', 'Smith', 40000, 'Cashier', 3905),
 (1021, 'William', 'Jones', 62000, 'Manager', 3903),
 (1022, 'Ella', 'Davis', 42000, 'Assistant Manager', 3903),
 (1023, 'James', 'Taylor', 50000, 'Sales Associate', 3902),
 (1024, 'Aiden', 'Hill', 75000, 'Marketing Coordinator', 3903),
 (1025, 'Lily', 'Moore', 48000, 'Stock Clerk', 3904),
 (1026, 'Benjamin', 'Wilson', 38000, 'Sales Associate', 3905),
 (1027, 'Zoe', 'Ward', 57000, 'Manager', 3904),
 (1028, 'Henry', 'Brown', 42000, 'Assistant Manager', 3904),
 (1029, 'Grace', 'Allen', 52000, 'Sales Associate', 3902),
 (1030, 'Jackson', 'Perez', 73000, 'Cashier', 3903),
 (1031, 'Madison', 'Carter', 46000, 'Marketing Coordinator', 3904),
 (1032, 'Lucas', 'Thomas', 40000, 'Stock Clerk', 3905),
 (1033, 'Chloe', 'Baker', 36000, 'Cashier', 3906),
 (1034, 'Logan', 'Gonzalez', 59000, 'Manager', 3905),
 (1035, 'Avery', 'Fisher', 42000, 'Assistant Manager', 3905),
 (1036, 'Elijah', 'Hall', 51000, 'Sales Associate', 3903),
 (1037, 'Scarlett', 'Parker', 34000, 'Customer Service Representative', 3904),
 (1038, 'Mason', 'Cruz', 43000, 'Cashier', 3905),
 (1039, 'Addison', 'Morgan', 67000, 'Manager', 3906),
 (1040, 'Aubrey', 'Butler', 38000, 'Assistant Manager', 3906),
 (1041, 'Liam', 'Barnes', 49000, 'Sales Associate', 3902),
 (1042, 'Evelyn', 'Cooper', 72000, 'Marketing Coordinator', 3903),
 (1043, 'Noah', 'Harrison', 42000, 'Stock Clerk', 3904),
 (1044, 'Aria', 'Reed', 36000, 'Sales Associate', 3905),
 (1045, 'Sebastian', 'Sullivan', 60000, 'Manager', 3907),
 (1046, 'Ryan', 'Anderson', 45000, 'Assistant Manager', 3903),
 (1047, 'Sophia', 'Evans', 40000, 'Assistant Manager', 3907),
 (1048, 'Ethan', 'Reed', 35000, 'Customer Service Representative', 3907),
 (1049, 'Isabella', 'Gonzalez', 32000, 'Cashier', 3907),
 (1050, 'Alexander', 'Fisher', 55000, 'Sales Associate', 3907),
 (1051, 'Olivia', 'Jones', 70000, 'Marketing Coordinator', 3907),
 (1052, 'Aiden', 'Barnes', 75000, 'Manager', 3908),
 (1053, 'Emma', 'Hill', 42000, 'Assistant Manager', 3908),
 (1054, 'Mia', 'Ward', 50000, 'Sales Associate', 3908),
 (1055, 'Lucas', 'Brown', 32000, 'Cashier', 3908),
 (1056, 'Sophie', 'Clark', 55000, 'Sales Associate', 3908),
 (1057, 'Liam', 'Perez', 73000, 'Cashier', 3909),
 (1058, 'Chloe', 'Carter', 46000, 'Marketing Coordinator', 3909),
 (1059, 'Noah', 'Thomas', 40000, 'Stock Clerk', 3909),

```

(1060, 'Aria', 'Baker', 36000, 'Sales Associate', 3909),
(1061, 'Sebastian', 'Moore', 32000, 'Cashier', 3909),
(1062, 'Ryan', 'Fisher', 45000, 'Assistant Manager', 3910),
(1063, 'Sophia', 'Garcia', 35000, 'Customer Service Representative', 3910),
(1064, 'Ethan', 'Wilson', 32000, 'Cashier', 3910),
(1065, 'Isabella', 'Clark', 55000, 'Sales Associate', 3910),
(1066, 'Alexander', 'Davis', 70000, 'Marketing Coordinator', 3910);

```

```

INSERT INTO Receipt (receiptid, totalamount, date, paymentmethod, storeid, employeeid) VALUES
(258369, '32.48', '2023-12-15 12:37:50+01', 'CARD', 3901, 1016),
(456287, '21.73', '2023-12-14 20:33:32+01', 'CASH', 3905, 1038),
(987654, '899.99', '2023-12-16 14:45:20+01', 'CARD', 3901, 1016),
(123789, '11.60', '2023-12-14 18:20:45+01', 'CASH', 3908, 1055),
(654321, '41.25', '2023-12-17 10:10:05+01', 'CARD', 3907, 1049),
(789456, '16.34', '2023-12-15 08:55:30+01', 'CASH', 3902, 1002),
(987123, '13.47', '2023-12-18 15:20:10+01', 'CASH', 3903, 1012),
(456789, '29.15', '2023-12-16 09:45:55+01', 'CARD', 3906, 1033),
(654987, '7.99', '2023-12-14 14:30:25+01', 'CASH', 3909, 1061),
(789321, '45.60', '2023-12-17 17:12:40+01', 'CARD', 3904, 1017),
(123456, '18.75', '2023-12-15 11:05:15+01', 'CARD', 3908, 1055),
(321789, '62.30', '2023-12-18 13:40:30+01', 'CASH', 3907, 1049),
(456123, '33.75', '2023-12-14 20:05:45+01', 'CARD', 3905, 1038),
(987321, '25.60', '2023-12-17 08:30:00+01', 'CARD', 3902, 1002),
(987456, '14.20', '2023-12-16 12:18:55+01', 'CASH', 3901, 1016),
(654123, '8.99', '2023-12-15 16:55:20+01', 'CARD', 3903, 1012),
(123987, '19.35', '2023-12-17 21:40:10+01', 'CASH', 3906, 1033),
(789654, '37.50', '2023-12-18 10:22:35+01', 'CARD', 3908, 1055),
(321456, '28.60', '2023-12-16 14:50:50+01', 'CASH', 3904, 1017),
(456321, '9.75', '2023-12-15 18:33:15+01', 'CARD', 3905, 1038),
(653421, '20.45', '2023-12-17 12:05:40+01', 'CASH', 3907, 1049),
(987789, '51.80', '2023-12-14 09:40:55+01', 'CARD', 3909, 1061),
(123321, '16.99', '2023-12-18 16:20:20+01', 'CASH', 3902, 1002),
(789987, '22.50', '2023-12-15 22:15:45+01', 'CARD', 3901, 1016),
(321987, '11.80', '2023-12-17 19:50:10+01', 'CASH', 3903, 1012),
(981026, '41.25', '2023-12-16 11:33:35+01', 'CARD', 3906, 1033),
(650187, '14.60', '2023-12-14 17:08:00+01', 'CASH', 3904, 1017),
(789123, '35.90', '2023-12-16 08:15:50+01', 'CASH', 3907, 1049),
(321654, '18.45', '2023-12-15 10:00:15+01', 'CARD', 3908, 1055),
(456089, '9.99', '2023-12-17 14:45:40+01', 'CASH', 3909, 1061),
(784956, '17.50', '2023-12-17 12:40:20+01', 'CASH', 3906, 1033),
(876543, '16.25', '2023-12-05 12:15:20+01', 'CASH', 3902, 1002),
(765432, '25.00', '2023-11-15 15:45:30+01', 'CARD', 3903, 1012),
(543219, '8.99', '2023-10-25 09:10:05+01', 'CARD', 3905, 1038),
(432198, '22.65', '2023-10-26 13:55:30+01', 'CASH', 3906, 1033),
(219876, '9.35', '2023-09-06 09:45:55+01', 'CASH', 3908, 1055),
(109876, '22.99', '2023-08-15 14:30:25+01', 'CARD', 3909, 1061),
(654322, '18.25', '2022-11-30 20:05:45+01', 'CARD', 3904, 1017),
(543220, '9.60', '2022-11-29 08:30:00+01', 'CASH', 3905, 1038),
(432199, '15.99', '2022-10-10 12:18:55+01', 'CARD', 3906, 1033),
(321988, '5.45', '2022-10-09 16:55:20+01', 'CASH', 3907, 1049),
(219877, '12.20', '2022-09-20 21:40:10+01', 'CARD', 3908, 1055),
(109877, '19.50', '2022-09-19 10:22:35+01', 'CASH', 3909, 1061),

```



```
(876544, '8.75', '2022-08-27 18:33:15+01', 'CASH', 3902, 1002);
```

```
INSERT INTO Customer (email, name, surname, gender, password, dateofbirth, loyaltypoints,
storeid) VALUES
('juanhernandez@gmail.com', 'Juan', 'Hernandez', 'Male', 'Juan1234', '2003-01-15', 45, 3909),
('lorenzaGT@gmail.com', 'Lorenza', 'Grande', 'Female', 'Lorenza1234', '2000-12-10', 89, 3904),
('marco@email.it', 'Marco', 'Rossi', 'Male', 'Marco1234', '1992-05-08', 95, 3904),
('giulia@email.it', 'Giulia', 'Bianchi', 'Female', 'Giulia5678', '1985-12-14', 110, 3906),
('paolo@email.it', 'Paolo', 'Ferrari', 'Male', 'Paolo9876', '1980-09-20', 80, 3902),
('carla@gmail.com', 'Carla', 'Garcia', 'Female', 'Carla1234', '1995-07-22', 120, 3901),
('aysefatma@hotmail.com', 'Ayse', 'Fatma', 'Female', 'Aysef5678', '1988-04-18', 75, 3907),
('lucia@yahoo.com', 'Lucia', 'Fernandez', 'Other', 'Lucia9876', '1990-11-30', 50, 3903),
('john_doe@gmail.com', 'John', 'Doe', 'Male', 'JohnDoe123', '1975-08-25', 60, 3905),
('jane_smith@gmail.com', 'Jane', 'Smith', 'Female', 'JaneSmith456', '1970-03-12', 70,
3908),
('robert_smith@gmail.com', 'Robert', 'Smith', 'Male', 'RobertSmith789', '1982-06-28', 95,
3902),
('emily_jones@gmail.com', 'Emily', 'Jones', 'Female', 'EmilyJones987', '1987-09-15', 110,
3906),
('michael_brown@gmail.com', 'Michael', 'Brown', 'Male', 'MichaelBrown654', '1991-12-03',
75, 3907),
('susan_wilson@gmail.com', 'Susan', 'Wilson', 'Female', 'SusanWilson321', '1984-05-20',
85, 3909),
('david_clark@gmail.com', 'David', 'Clark', 'Male', 'DavidClark234', '1978-02-18', 100,
3901);
```

```
INSERT INTO CONTAINS(receiptid, productid, quantity)
VALUES
```

```
(258369, 07004, 2), --2,49*2
(789456, 12002, 1), --5,55
(123789, 10004, 1), --2,80
(123789, 11002, 2), --4,40*2
(789456, 14003, 1), --10,79
(654321, 06001, 3), --2,29*3
(654321, 01002, 1), --3,0
(258369, 15002, 1), --25,0
(258369, 02004, 1), --2,50
(987654, 15005, 1), --899,99
(456287, 10005, 1), --2,54
(654321, 04003, 2), --15,69*2
(456287, 10004, 3), --2,80*3
(456287, 14003, 1), --10,79
(987123, 05001, 2), -- 5.49 * 2
(456789, 08003, 1), -- 2.25
(654987, 07003, 2), -- 3.35 * 2
(789321, 13001, 1), -- 4.79
(123456, 03004, 3), -- 2.29 * 3
(321789, 02001, 1), -- 2.0
(456123, 10003, 1), -- 2.0
(987321, 06004, 2), -- 1.79 * 2
```

```

(987456, 13003, 1), -- 3.15
(654321, 15004, 1), -- 599.99
(123987, 08002, 2), -- 3.19 * 2
(789654, 10001, 3), -- 2.50 * 3
(321456, 02003, 1), -- 2.75
(456321, 09003, 1), -- 5.39
(654321, 10002, 1), -- 2.40
(987789, 14001, 2), -- 8.49 * 2
(123321, 09002, 1), -- 4.75
(789987, 07002, 3), -- 3.90 * 3
(321987, 04001, 2), -- 8.89 * 2
(987456, 01004, 1), -- 15.99
(654987, 11001, 1), -- 7.29
(123456, 06001, 2), -- 5.99 * 2
(789123, 05003, 1), -- 4.25
(321654, 12003, 1), -- 9.75
(456789, 03002, 2), -- 3.49 * 2
(654321, 07001, 1), -- 6.49
(987654, 12001, 1), -- 11.99
(123789, 01003, 1), -- 19.99
(789456, 11002, 1); -- 4.40

```

```

INSERT INTO BelongsTo(receiptid, email) VALUES
(258369, 'lucia@yahoo.com'),
(987654, 'susan_wilson@gmail.com'),
(123789, 'marco@email.it'),
(789456, 'juanhernandez@gmail.com'),
(456287, 'giulia@email.it'),
(654321, 'aysefatma@hotmail.com'),
(987123, 'lucia@yahoo.com'),
(654987, 'jane_smith@gmail.com'),
(789321, 'paolo@email.it'),
(123456, 'lorenzaGT@gmail.com'),
(321789, 'aysefatma@hotmail.com'),
(456123, 'marco@email.it'),
(987321, 'lucia@yahoo.com'),
(987456, 'lorenzaGT@gmail.com'),
(654123, 'carla@gmail.com'),
(123987, 'paolo@email.it'),
(789654, 'susan_wilson@gmail.com'),
(321456, 'juanhernandez@gmail.com'),
(456321, 'lucia@yahoo.com'),
(987789, 'carla@gmail.com'),
(123321, 'lorenzaGT@gmail.com'),
(789987, 'jane_smith@gmail.com'),
(321987, 'marco@email.it'),
(789123, 'carla@gmail.com'),
(321654, 'jane_smith@gmail.com'),
(456789, 'lorenzaGT@gmail.com');

```

Principal Queries

In this section we report the queries needed to perform some operations on the database:

1. For each store, get its id, the list of products that need to be bought (id and name) and the minimum quantity
2. For each customer in the loyalty program, get its email, the list of products that currently have a promotion on their preferred store, the discount info and the start and end dates of the promotion
3. For each cashier, get their surname and name, the id of the store they work for and how many times has a customer used the loyalty program when they were attending at the register, and how many of them were unique customers
4. For each store, get its id and the percentage of products bought ordered by category during the current month
5. Obtain the category from which women between 40 and 60 years old buy the most products (and quantity)

```
SELECT s.storeid AS store_id, p.productid AS product_id, p.name AS product_name,
COALESCE(s.reorderlevel - (s.stockquantity + COALESCE(SUM(sp.quantity), 0)), 0) AS
min_quantity_to_order
FROM Stores AS s INNER JOIN Product AS p
ON s.productid = p.productid
LEFT JOIN Supplies AS sp
ON s.productid = sp.productid AND s.storeid = sp.storeid
GROUP BY s.storeid, p.productid, s.reorderlevel, s.stockquantity
HAVING s.reorderlevel > (s.stockquantity + COALESCE(SUM(sp.quantity), 0));
```

	store_id integer	product_id integer	product_name character varying (250)	min_quantity_to_order bigint
1	3903	11003	Clean Sweep Broom	4
2	3907	8001	Lays Potato Chips	17
3	3905	6001	NatureHarbor Organic Canned Beans	17

```
SELECT c.email AS customer_email, p.name AS product_name, pr.discountinfo AS
discount_info, pr.startenddates AS promotion_dates
FROM Customer c INNER JOIN Store AS s
ON c.storeid = s.storeid
INNER JOIN Promotion AS pr
ON s.storeid = pr.storeid
INNER JOIN Product AS p
ON pr.productid = p.productid
WHERE pr.startenddates @> CURRENT_DATE;
```

	customer_email character varying (254)	product_name character varying (250)	discount_info character varying (250)	promotion_dates daterange
1	juanhernandez@gmail.com	Blitz Blender	20% off on Blitz Blender	[2023-12-01,2024-02-29)
2	juanhernandez@gmail.com	Loreal Paris Shampoo	15% off on Loreal Paris Shampoo	[2023-12-10,2024-03-16)
3	juanhernandez@gmail.com	Lays Potato Chips	25% off on Lays Potato Chips	[2023-12-01,2024-01-01)
4	lorenzaGT@gmail.com	NatureHarbor Organic Canned Beans	20% off on NatureHarbor Organic Canned Beans	[2023-12-12,2024-01-29)
5	marco@email.it	NatureHarbor Organic Canned Beans	20% off on NatureHarbor Organic Canned Beans	[2023-12-12,2024-01-29)
6	giulia@email.it	Lays Potato Chips	12% off on Lays Potato Chips	[2023-11-05,2023-12-29)
7	giulia@email.it	DeliMaster Smoked Chicken Breast	30% off on DeliMaster Smoked Chicken Breast	[2023-12-10,2023-12-18)
8	aysefatma@hotmail.com	Blitz Blender	12% off on Blitz Blender	[2023-11-05,2023-12-29)
9	lucia@yahoo.com	TechCo Smartphone X1	20% off on TechCo Smartphone X1	[2023-10-01,2024-01-01)
10	jane_smith@gmail.com	Whoole Milk	15% off on Whoole Milk	[2023-11-15,2024-02-16)
11	emily_jones@gmail.com	Lays Potato Chips	12% off on Lays Potato Chips	[2023-11-05,2023-12-29)
12	emily_jones@gmail.com	DeliMaster Smoked Chicken Breast	30% off on DeliMaster Smoked Chicken Breast	[2023-12-10,2023-12-18)
13	michael_brown@gmail.com	Blitz Blender	12% off on Blitz Blender	[2023-11-05,2023-12-29)
14	susan_wilson@gmail.com	Blitz Blender	20% off on Blitz Blender	[2023-12-01,2024-02-29)
15	susan_wilson@gmail.com	Loreal Paris Shampoo	15% off on Loreal Paris Shampoo	[2023-12-10,2024-03-16)
16	susan_wilson@gmail.com	Lays Potato Chips	25% off on Lays Potato Chips	[2023-12-01,2024-01-01)

```

SELECT storeid, surname, name, total_receipts, unique_customers
FROM (SELECT e.surname, e.name, e.storeid, COUNT(r.receiptid) AS total_receipts,
COUNT(DISTINCT b.email) AS unique_customers
FROM Employee AS e INNER JOIN Receipt AS r
ON e.employeeid = r.employeeid
LEFT JOIN BelongsTo AS b
ON r.receiptid = b.receiptid
WHERE e.position = 'Cashier'
GROUP BY e.surname, e.name, e.storeid)
ORDER BY storeid;

```

	storeid integer	surname character varying (250)	name character varying (250)	total_receipts bigint	unique_customers bigint
1	3901	Moore	Jessica	4	4
2	3902	Smith	Paula	5	3
3	3903	Miller	Sophie	4	3
4	3904	Wilson	David	4	2
5	3905	Cruz	Mason	5	3
6	3906	Baker	Chloe	6	2
7	3907	Gonzalez	Isabella	5	2
8	3908	Brown	Lucas	6	4
9	3909	Moore	Sebastian	5	2

```

SELECT store_id, category_name, ROUND(100*category_quantity/total_quantity,2) AS percentage
FROM (SELECT s.storeid AS store_id, c.categoryname AS category_name,
SUM(COALESCE(co.quantity, 0)) AS category_quantity,
SUM(SUM(COALESCE(co.quantity, 0))) OVER (PARTITION BY s.storeid) AS total_quantity
FROM Category AS c LEFT JOIN Product AS p
ON c.categoryname = p.category
LEFT JOIN Contains AS co
ON p.productid = co.productid
LEFT JOIN Receipt AS r
ON co.receiptid = r.receiptid
LEFT JOIN Store AS s
ON r.storeid = s.storeid
WHERE DATE_TRUNC('month', r.date) = DATE_TRUNC('month', CURRENT_DATE)
GROUP BY s.storeid, c.categoryname);

```

	store_id integer	category_name character varying (250)	percentage numeric
1	3901	Bakery and Pastries	9.09
2	3901	Beverages	45.45
3	3901	Electronics and Appliances	18.18
4	3901	Fresh Produce	9.09
5	3901	Health and Beauty	9.09
6	3901	Pet Supplies	9.09
7	3902	Baby Care	16.67
8	3902	Cereal and Breakfast Foods	16.67
9	3902	Grocery and Staples	33.33
10	3902	Health and Beauty	16.67
11	3902	Household and Cleaning Supplies	16.67
12	3903	Frozen Foods	50.00
13	3903	Meat and Seafood	50.00
14	3904	Bakery and Pastries	50.00
15	3904	Pet Supplies	50.00
16	3905	Baby Care	14.29
17	3905	Cereal and Breakfast Foods	14.29
18	3905	Condiments and Sauces	71.43
19	3906	Dairy and Eggs	40.00
20	3906	Snacks	60.00

21	3907	Bakery and Pastries	9.09
22	3907	Beverages	9.09
23	3907	Condiments and Sauces	9.09
24	3907	Electronics and Appliances	9.09
25	3907	Fresh Produce	9.09
26	3907	Frozen Foods	9.09
27	3907	Grocery and Staples	27.27
28	3907	Meat and Seafood	18.18
29	3908	Condiments and Sauces	30.77
30	3908	Dairy and Eggs	23.08
31	3908	Fresh Produce	7.69
32	3908	Grocery and Staples	15.38
33	3908	Health and Beauty	7.69
34	3908	Household and Cleaning Supplies	15.38
35	3909	Baby Care	40.00
36	3909	Beverages	40.00
37	3909	Household and Cleaning Supplies	20.00

```



SELECT categoryname, max_quantity
FROM (SELECT p.category AS categoryname, SUM(co.quantity) AS max_quantity
      FROM Contains AS co INNER JOIN Product AS p
        ON co.productid = p.productid
      INNER JOIN Receipt AS r
        ON co.receiptid = r.receiptid
      INNER JOIN BelongsTo AS b
        ON co.receiptid = b.receiptid
      INNER JOIN Customer AS c
        ON b.email = c.email
      WHERE gender = 'Female' AND dateofbirth BETWEEN CURRENT_DATE - INTERVAL '60
years' AND CURRENT_DATE - INTERVAL '40 years'
      GROUP BY p.category)
WHERE max_quantity = (SELECT MAX(total_quantity) AS max_quantity
                     FROM (SELECT p.category AS categoryname, SUM(co.quantity)
                           AS total_quantity
                           FROM Contains AS co INNER JOIN Product AS p
                             ON co.productid = p.productid
                           INNER JOIN Receipt AS r
                             ON co.receiptid = r.receiptid
                           INNER JOIN BelongsTo AS b
                             ON co.receiptid = b.receiptid
                           INNER JOIN Customer AS c
                             ON b.email = c.email
                           WHERE gender = 'Female' AND dateofbirth BETWEEN

```

```

CURRENT_DATE - INTERVAL '60
years' AND CURRENT_DATE - INTERVAL '40 years'
GROUP BY p.category));

```

	categoryname character varying (250) 	max_quantity bigint 
1	Beverages	5

JDBC Implementations of the Principal Queries and Visualization

In the following, we present a Java class that reads data from the database and displays the results on the screen.

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.PreparedStatement;
public class ecommesta {

    // The JDBC driver to be used
    private static final String Driver = "org.postgresql.Driver";

    // The URL of the database to be accessed
    private static final String Database = "jdbc:postgresql://localhost:5432/Mercadone";

    // Username for accessing the database
    private static final String User = "postgres";

    // Password for accessing the database
    private static final String Password = "ecommesta";

    public static void main(String[] args) {

        // Connection to the DBMS
        Connection con = null;

        // Statements to be executed
        Statement st1 = null;
        Statement st2 = null;

        // Results of the statements
        ResultSet resset1 = null;
        ResultSet resset2 = null;

        // Start and end times of a statement
        long start;

```

```

long end;

try {
    // Register the JDBC driver
    Class.forName(Driver);
    System.out.printf("Driver %s is successfully registered. %n", Driver);

} catch (ClassNotFoundException e) {
    System.out.printf("Driver %s not found : %s.%n", Driver, e.getMessage());
    // terminate with a generic error code
    System.exit(-1);
}

try {
    // Connect to the database
    start = System.currentTimeMillis();
    con = DriverManager.getConnection(Database, User, Password);
    end = System.currentTimeMillis();
    System.out.printf("Connection to database %s successfully established in %,d
milliseconds %n", Database, end - start);

    // Create first statement to execute the first query
    start = System.currentTimeMillis();
    st1 = con.createStatement();
    end = System.currentTimeMillis();
    System.out.printf("%nStatement 1 Description: For each cashier, get their surname
and name, the id of the store they work for %nand how many times has a customer
used the loyalty program when they were attending at the register, %nand how many
of them were unique customers. %n");
    System.out.printf("%nStatement 1 successfully created in %,d milliseconds.", end - start);

    // Execute the query and get the results
    String sqlqr1 =
        "SELECT storeid, surname, name, total_receipts, unique_customers\n" +
        "    FROM (SELECT e.surname, e.name, e.storeid, COUNT(r.receiptid)
AS total_receipts,\n" +
        "        COUNT(DISTINCT b.email) AS unique_customers\n" +
        "            FROM Employee AS e INNER JOIN Receipt AS r\n" +
        "                ON e.employeeid = r.employeeid\n" +
        "            LEFT JOIN BelongsTo AS b\n" +
        "                ON r.receiptid = b.receiptid\n" +
        "            WHERE e.position = 'Cashier'\n" +
        "            GROUP BY e.surname, e.name, e.storeid)\n" +
        "ORDER BY storeid;";
    start = System.currentTimeMillis();
    resset1 = st1.executeQuery(sqlqr1);
    end = System.currentTimeMillis();

    System.out.printf("%nQUERY1: Query successfully executed %,d milliseconds.%n", end - start);
    System.out.printf("Query 1 Results: %n");
    System.out.printf("-----%n");
    System.out.printf("| %-7s | %-8s | %-9s | %-14s | %-16s |%n", "StoreID", "Surname", "Name",
    "Total Receipts", "Unique Customers");
    System.out.printf("-----%n");
    int storeid;

```



```

String surname;
String name;
int total_receipts;
int unique_customers;

// print the results via a loop
while (resset1.next()) {
    storeid = resset1.getInt("storeid");
    surname = resset1.getString("surname");
    name = resset1.getString("name");
    total_receipts = resset1.getInt("total_receipts");
    unique_customers = resset1.getInt("unique_customers");
    System.out.printf("| %-7s | %-8s | %-9s | %-7s | %-9s |%n", storeid,
        surname, name, total_receipts, unique_customers);
}

// Create second statement to execute the second query
String sqlqr2 =
    "SELECT store_id, category_name, ROUND(100*category_quantity/total_quantity,2)
    AS percentage\n" +
    "    FROM (SELECT s.storeid AS store_id, c.categoryname AS category_name,\n" +
    "        SUM(COALESCE(co.quantity, 0)) AS category_quantity,\n" +
    "        SUM(SUM(COALESCE(co.quantity, 0))) OVER (PARTITION BY s.storeid)
    AS total_quantity\n" +
    "    FROM Category AS c LEFT JOIN Product AS p\n" +
    "        ON c.categoryname = p.category\n" +
    "    LEFT JOIN Contains AS co\n" +
    "        ON p.productid = co.productid\n" +
    "    LEFT JOIN Receipt AS r\n" +
    "        ON co.receiptid = r.receiptid\n" +
    "    LEFT JOIN Store AS s\n" +
    "        ON r.storeid = s.storeid\n" +
    "    WHERE DATE_TRUNC('month', r.date) = DATE_TRUNC('month', CURRENT_DATE)\n" +
    "    GROUP BY s.storeid, c.categoryname);";

start = System.currentTimeMillis();
st2 = con.createStatement();
end = System.currentTimeMillis();
System.out.printf("%nStatement 2 Description: For each store, get its id and the
percentage of products %nbought ordered by category during the current month. %n");
System.out.printf("%nStatement 2 successfully created in %,d milliseconds.", end - start);

// Execute second query
start = System.currentTimeMillis();
resset2 = st2.executeQuery(sqlqr2);
end = System.currentTimeMillis();
System.out.printf("%nQUERY2: Query successfully executed %,d milliseconds.%n", end - start);

System.out.printf("Query 2 Results: %n");
System.out.printf("-----%n");
System.out.printf("| %-7s | %-32s | %-10s |%n", "StoreID", "Category Name", "Percentage");
System.out.printf("-----%n");
int storeid2;
String category_name;
double percentage;

```

```

        while (reset2.next()) {
            storeid2 = reset2.getInt("store_id");
            category_name = reset2.getString("category_name");
            percentage = reset2.getDouble("percentage");

            System.out.printf("| %-7s | %-32s | %-10s |%n", storeid2, category_name, percentage);
        }

    } catch (SQLException e) {
        System.out.printf("Database access error: %n");
        // cycle in the exception chain
        while (e != null) {
            //e.printStackTrace();
            System.out.printf("-Message: %s", e.getMessage());
            System.out.printf("-SQL status code: %s", e.getSQLState());
            System.out.printf("-SQL error code: %s", e.getErrorCode());
            System.out.printf("%n");
            e = e.getNextException();
        }
    }

    finally {
        try {
            if (reset1 != null || reset2 != null) {
                start = System.currentTimeMillis();
                reset1.close();
                reset2.close();
                end = System.currentTimeMillis();
                System.out.printf("%nResult sets are successfully closed in final block in %,d milliseconds.", end - start);
            }
            if (st1 != null || st2 != null) {
                start = System.currentTimeMillis();
                st1.close();
                st2.close();
                end = System.currentTimeMillis();
                System.out.printf("%nStatements are successfully closed in final block in %,d milliseconds.", end - start);
            }
            if (con != null) {
                start = System.currentTimeMillis();
                con.close();
                end = System.currentTimeMillis();
                System.out.printf("%nConnection successfully closed in final block in %,d milliseconds. %n", end - start);
            }
        }

        } catch (SQLException e) {
            System.out.printf("Error while releasing resources: %s", e.getMessage());
            while (e != null) {
                //e.printStackTrace();
                System.out.printf("-Message: %s", e.getMessage());
                System.out.printf("-SQL status code: %s", e.getSQLState());
            }
        }
    }
}

```

```

        System.out.printf("-SQL error code: %s", e.getErrorCode());
        System.out.printf("%n");
        e = e.getNextException();
    }

    } finally{
        resset1 = null;
        resset2 = null;
        st2 = null;
        st1 = null;
        con = null;
        System.out.printf("Resources are released to the garbage collector. %n");
    }
}
System.out.printf("Program ended. %n");
}
}

```

```

Driver org.postgresql.Driver is successfully registered.
Connection to database jdbc:postgresql://localhost:5432/Mercadone successfully established in 909 milliseconds .

Statement 1 Description: For each cashier, get their surname and name, the id of the store they work for
and how many times has a customer used the loyalty program when they were attending at the register,
and how many of them were unique customers.

Statement 1 successfully created in 8 milliseconds.
QUERY1: Query successfully executed 16 milliseconds.
Query 1 Results:
-----
| StoreID | Surname | Name      | Total Receipts | Unique Customers |
-----
| 3901    | Moore   | Jessica   | 4              | 4                |
| 3902    | Smith   | Paula     | 5              | 3                |
| 3903    | Miller  | Sophie    | 4              | 3                |
| 3904    | Wilson  | David     | 4              | 2                |
| 3905    | Cruz    | Mason     | 5              | 3                |
| 3906    | Baker   | Chloe     | 6              | 2                |
| 3907    | Gonzalez | Isabella  | 5              | 2                |
| 3908    | Brown   | Lucas     | 6              | 4                |
| 3909    | Moore   | Sebastian | 5              | 2                |

```

```

Statement 2 successfully created in 0 milliseconds.
QUERY2: Query successfully executed 8 milliseconds.
Query 2 Results:
-----
| StoreID | Category Name          | Percentage |
-----
| 3901    | Bakery and Pastries    | 9.09      |
| 3901    | Beverages              | 45.45     |
| 3901    | Electronics and Appliances | 18.18    |
| 3901    | Fresh Produce          | 9.09      |
| 3901    | Health and Beauty      | 9.09      |
| 3901    | Pet Supplies           | 9.09      |
| 3902    | Baby Care              | 16.67     |
| 3902    | Cereal and Breakfast Foods | 16.67    |
| 3902    | Grocery and Staples    | 33.33     |
| 3902    | Health and Beauty      | 16.67     |
| 3902    | Household and Cleaning Supplies | 16.67  |
| 3903    | Frozen Foods           | 50.0      |
| 3903    | Meat and Seafood       | 50.0      |
| 3904    | Bakery and Pastries    | 50.0      |
| 3904    | Pet Supplies           | 50.0      |
| 3905    | Baby Care              | 14.29     |
| 3905    | Cereal and Breakfast Foods | 14.29    |
| 3905    | Condiments and Sauces  | 71.43     |
-----
| 3906    | Dairy and Eggs         | 40.0      |
| 3906    | Snacks                 | 60.0      |
| 3907    | Bakery and Pastries    | 9.09      |
| 3907    | Beverages              | 9.09      |
| 3907    | Condiments and Sauces  | 9.09      |
| 3907    | Electronics and Appliances | 9.09    |
| 3907    | Fresh Produce          | 9.09      |
| 3907    | Frozen Foods           | 9.09      |
| 3907    | Grocery and Staples    | 27.27     |
| 3907    | Meat and Seafood       | 18.18     |
| 3908    | Condiments and Sauces  | 30.77     |
| 3908    | Dairy and Eggs         | 23.08     |
| 3908    | Fresh Produce          | 7.69      |
| 3908    | Grocery and Staples    | 15.38     |
| 3908    | Health and Beauty      | 7.69      |
| 3908    | Household and Cleaning Supplies | 15.38  |
| 3909    | Baby Care              | 40.0      |
| 3909    | Beverages              | 40.0      |
| 3909    | Household and Cleaning Supplies | 20.0    |
-----

Result sets are successfully closed in final block in 1 milliseconds.
Statements are successfully closed in final block in 0 milliseconds.
Connection successfully closed in final block in 1 milliseconds.
Resources are released to the garbage collector.
Program ended.

```

Group Members Contribution

- **Umut Berk CAKMAKCI:** Contributed to the development of **Populate the Database** and **JDBC Implementations** sections;
- **Alara Selen OZGEN:** Responsible for the development of **Populate the Database: Example** sections;
- **Meltem YANOGLU:** Contributed to the development of **Populate the Database** and **JDBC Implementations** sections;
- **Isabel TORRES-PARDO LÓPEZ:** Responsible for developing the sections of the project pertaining to **Variations to the Relational Schema** and **Populate the Database: Example**;
- **Alejandro VARELA MARTÍN:** Responsible for developing the sections of the project pertaining to **Physical Schema** and **Principal Queries**;
- **Desalegn MATHEWOS MALOCHE:** Responsible for **from 'populate database example' employee part**
- **Milad Faghih Loo:** We could not make contact with him. He did not participate any work.

Each member of the group (except Milad) has diligently reviewed the text, offering insightful comments and implementing necessary corrections to enhance its quality and accuracy.