DROP TABLE IF EXISTS FOLLOWS\_TABLE;

DROP TABLE IF EXISTS SHOWS;

DROP TABLE IF EXISTS TRACKS\_PRODUCT\_ORDER\_EVENT;

DROP TABLE IF EXISTS ORDER\_EVENT;

DROP TABLE IF EXISTS SHIPPED\_BY;

DROP TABLE IF EXISTS ORDER\_HISTORY;

DROP TABLE IF EXISTS SHIPMENT;

DROP TABLE IF EXISTS APPLIES\_CART\_DISCOUNT;

DROP TABLE IF EXISTS SELLS;

DROP TABLE IF EXISTS FAVORİTES;

DROP TABLE IF EXISTS APPLIES\_STORE\_DISCOUNT;

DROP TABLE IF EXISTS ANSWER;

DROP TABLE IF EXISTS QUESTION;

DROP TABLE IF EXISTS LISTS;

DROP TABLE IF EXISTS STORE;

DROP TABLE IF EXISTS HISTORY;

DROP TABLE IF EXISTS PRODUCT\_REVIEW\_IMAGES;

DROP TABLE IF EXISTS SELLER\_REVIEW;

DROP TABLE IF EXISTS PRODUCT\_REVIEW;

DROP TABLE IF EXISTS VIEWED\_BY ;

DROP TABLE IF EXISTS VISITOR;

DROP TABLE IF EXISTS SELLER;

DROP TABLE IF EXISTS PICK\_UP\_POINT;

DROP TABLE IF EXISTS PRODUCT\_IMAGES;

DROP TABLE IF EXISTS APPLIES\_BRAND\_DISCOUNT;

DROP TABLE IF EXISTS CAN\_CATEGORIZED\_AS;

DROP TABLE IF EXISTS APPLIES\_PRODUCT\_DISCOUNT;

DROP TABLE IF EXISTS NOTIFICATION\_PREFERENCE;

DROP TABLE IF EXISTS REVIEW;

DROP TABLE IF EXISTS CART\_ITEMS;

DROP TABLE IF EXISTS MY\_ADDRESS;

DROP TABLE IF EXISTS SHIPPING\_DETAILS;

DROP TABLE IF EXISTS ORDERTABLE;

DROP TABLE IF EXISTS CART;

DROP TABLE IF EXISTS PAYMENT\_METHOD;

DROP TABLE IF EXISTS BUYER;

DROP TABLE IF EXISTS REGISTERED\_USER;

DROP TABLE IF EXISTS USERTABLE;

DROP TABLE IF EXISTS PRODUCT;

DROP TABLE IF EXISTS BRAND;

DROP TABLE IF EXISTS CATEGORY;

DROP TABLE IF EXISTS DISCOUNT;

DROP TABLE IF EXISTS PAGETABLE;

DROP TABLE IF EXISTS CARRIER;

--------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS USERTABLE (

user\_id INT PRIMARY KEY,

e\_mail VARCHAR(255) NOT NULL UNIQUE,

phone VARCHAR(20) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL,

reg\_date DATETIME NOT NULL,

lname VARCHAR(100),

fname VARCHAR(100)

);

CREATE TABLE IF NOT EXISTS BUYER (

buyer\_id INT PRIMARY KEY,

start\_date DATE NOT NULL,

end\_date DATE,

is\_standard\_member BOOLEAN DEFAULT TRUE,

is\_premium\_member BOOLEAN DEFAULT FALSE,

FOREIGN KEY (buyer\_id) REFERENCES USERTABLE(user\_id)

ON DELETE CASCADE

);

CREATE TABLE REGISTERED\_USER (

user\_id INT PRIMARY KEY,

FOREIGN KEY (user\_id) REFERENCES USERTABLE(user\_id)

ON DELETE CASCADE

);

--------------------------------------------------------------------------------

CREATE TABLE IF NOT EXISTS PAYMENT\_METHOD (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

buyer\_id INT NOT NULL,

payment\_type VARCHAR(50),

holder\_name VARCHAR(255),

isDefault BOOLEAN DEFAULT FALSE,

exp\_date DATE,

cvv VARCHAR(10),

google\_mail VARCHAR(255),

current\_balance DECIMAL(10,2) DEFAULT 0.00,

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE

);

-- Check constraints for PAYMENT\_METHOD

ALTER TABLE PAYMENT\_METHOD

ADD CONSTRAINT chk\_current\_balance CHECK (current\_balance >= 0);

ALTER TABLE PAYMENT\_METHOD

ADD CONSTRAINT chk\_cvv\_length CHECK (CHAR\_LENGTH(cvv) IN (3, 4));

CREATE TABLE IF NOT EXISTS CART (

cart\_id INT NOT NULL AUTO\_INCREMENT,

buyer\_id INT NOT NULL,

PRIMARY KEY (cart\_id, buyer\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS ORDERTABLE (

order\_id INT AUTO\_INCREMENT,

buyer\_id INT NOT NULL,

payment\_id INT UNIQUE,

total DECIMAL(10,2) NOT NULL CHECK (total > 0),

date DATETIME DEFAULT CURRENT\_TIMESTAMP,

cart\_id INT NOT NULL,

PRIMARY KEY (order\_id, buyer\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES CART(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (payment\_id) REFERENCES PAYMENT\_METHOD(payment\_id)

ON DELETE SET NULL,

FOREIGN KEY (cart\_id) REFERENCES CART(cart\_id)

ON DELETE CASCADE

);

CREATE TABLE SHIPPING\_DETAILS (

shipping\_id INT AUTO\_INCREMENT PRIMARY KEY,

buyer\_id INT NOT NULL,

order\_id INT NOT NULL,

phone VARCHAR(20) NOT NULL,

city VARCHAR(100) NOT NULL,

state VARCHAR(100) NOT NULL,

zip VARCHAR(20) NOT NULL,

country VARCHAR(100) NOT NULL,

street\_address VARCHAR(255) NOT NULL,

FOREIGN KEY (buyer\_id) REFERENCES ORDERTABLE(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES ORDERTABLE(order\_id)

ON DELETE CASCADE

);

-- Constraints for SHIPPING\_DETAILS

ALTER TABLE SHIPPING\_DETAILS

ADD CONSTRAINT chk\_zip\_numeric CHECK (zip REGEXP '^[0-9]+$');

ALTER TABLE SHIPPING\_DETAILS

ADD CONSTRAINT chk\_phone\_length CHECK (CHAR\_LENGTH(phone) >= 10);

CREATE TABLE BRAND (

brand\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL

);

CREATE TABLE PRODUCT (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

brand\_id INT,

review\_count INT DEFAULT 0,

name VARCHAR(255) NOT NULL,

is\_flash BOOLEAN DEFAULT FALSE,

descp TEXT,

stock\_level INT DEFAULT 0,

is\_best\_seller BOOLEAN DEFAULT FALSE,

price DECIMAL(10,2) NOT NULL,

size VARCHAR(50),

FOREIGN KEY (brand\_id) REFERENCES BRAND(brand\_id)

ON DELETE SET NULL

);

ALTER TABLE PRODUCT

ADD CONSTRAINT chk\_stock\_level CHECK (stock\_level >= 0);

ALTER TABLE PRODUCT

ADD CONSTRAINT chk\_price\_positive CHECK (price > 0);

CREATE TABLE CATEGORY (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

sub\_category\_id INT,

name VARCHAR(255) NOT NULL,

FOREIGN KEY (sub\_category\_id) REFERENCES CATEGORY(category\_id)

ON DELETE SET NULL

);

CREATE TABLE DISCOUNT (

discount\_id INT AUTO\_INCREMENT PRIMARY KEY,

min\_amount DECIMAL(10,2) DEFAULT 0.00,

type VARCHAR(50),

valid\_from DATE,

expiry\_date DATE,

amount DECIMAL(10,2) DEFAULT 0.00,

coupon\_code VARCHAR(255),

is\_standard BOOLEAN DEFAULT FALSE,

is\_coupon BOOLEAN DEFAULT FALSE

);

CREATE TABLE PAGETABLE (

page\_id INT AUTO\_INCREMENT PRIMARY KEY,

page\_url VARCHAR(255) NOT NULL

);

CREATE TABLE REVIEW (

review\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255),

rating INT CHECK (rating >= 1 AND rating <= 5),

buyer\_id INT,

is\_order\_delivered BOOLEAN DEFAULT FALSE,

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE SET NULL

);

CREATE TABLE NOTIFICATION\_PREFERENCE (

ntfctn\_pref\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

is\_email BOOLEAN DEFAULT TRUE,

is\_sms BOOLEAN DEFAULT FALSE,

is\_phone BOOLEAN DEFAULT FALSE,

FOREIGN KEY (user\_id) REFERENCES USERTABLE(user\_id)

ON DELETE CASCADE

);

CREATE TABLE CAN\_CATEGORIZED\_AS (

product\_id INT NOT NULL,

category\_id INT NOT NULL,

PRIMARY KEY (product\_id, category\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE,

FOREIGN KEY (category\_id) REFERENCES CATEGORY(category\_id)

ON DELETE CASCADE

);

CREATE TABLE APPLIES\_PRODUCT\_DISCOUNT (

discount\_id INT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (discount\_id, product\_id),

FOREIGN KEY (discount\_id) REFERENCES DISCOUNT(discount\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE APPLIES\_BRAND\_DISCOUNT (

discount\_id INT NOT NULL,

brand\_id INT NOT NULL,

PRIMARY KEY (discount\_id, brand\_id),

FOREIGN KEY (discount\_id) REFERENCES DISCOUNT(discount\_id)

ON DELETE CASCADE,

FOREIGN KEY (brand\_id) REFERENCES BRAND(brand\_id)

ON DELETE CASCADE

);

CREATE TABLE PRODUCT\_IMAGES (

product\_id INT NOT NULL,

image VARCHAR(255) NOT NULL,

PRIMARY KEY (product\_id, image),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE PICK\_UP\_POINT (

shipping\_id INT PRIMARY KEY,

point\_name VARCHAR(255),

open\_hours VARCHAR(255),

FOREIGN KEY (shipping\_id) REFERENCES SHIPPING\_DETAILS(shipping\_id)

ON DELETE CASCADE

);

CREATE TABLE MY\_ADDRESS (

shipping\_id INT PRIMARY KEY,

address\_name VARCHAR(255),

FOREIGN KEY (shipping\_id) REFERENCES SHIPPING\_DETAILS(shipping\_id)

ON DELETE CASCADE

);

CREATE TABLE CARRIER (

carrier\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

contact VARCHAR(255),

operational\_hours VARCHAR(255),

operating\_region VARCHAR(255),

operating\_country VARCHAR(255),

carrier\_type VARCHAR(50)

);

CREATE TABLE SELLER (

seller\_id INT PRIMARY KEY,

FOREIGN KEY (seller\_id) REFERENCES REGISTERED\_USER(user\_id)

ON DELETE CASCADE

);

CREATE TABLE SELLER\_REVIEW (

review\_id INT NOT NULL,

seller\_id INT NOT NULL,

PRIMARY KEY (review\_id),

FOREIGN KEY (review\_id) REFERENCES REVIEW(review\_id)

ON DELETE CASCADE,

FOREIGN KEY (seller\_id) REFERENCES SELLER(seller\_id)

ON DELETE CASCADE

);

CREATE TABLE PRODUCT\_REVIEW (

review\_id INT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (review\_id),

FOREIGN KEY (review\_id) REFERENCES REVIEW(review\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE VISITOR (

user\_id INT PRIMARY KEY,

session\_id VARCHAR(255),

ip\_address VARCHAR(45),

FOREIGN KEY (user\_id) REFERENCES USERTABLE(user\_id)

ON DELETE CASCADE

);

CREATE TABLE VIEWED\_BY (

page\_id INT NOT NULL,

user\_id INT NOT NULL,

PRIMARY KEY (page\_id, user\_id),

FOREIGN KEY (page\_id) REFERENCES PAGETABLE(page\_id)

ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES USERTABLE(user\_id)

ON DELETE CASCADE

);

CREATE TABLE PRODUCT\_REVIEW\_IMAGES (

review\_id INT NOT NULL,

image VARCHAR(255) NOT NULL,

PRIMARY KEY (review\_id, image),

FOREIGN KEY (review\_id) REFERENCES REVIEW(review\_id)

ON DELETE CASCADE

);

CREATE TABLE STORE (

store\_id INT NOT NULL AUTO\_INCREMENT,

seller\_id INT NOT NULL,

store\_name VARCHAR(255) NOT NULL,

follower\_amount INT DEFAULT 0,

PRIMARY KEY (store\_id, seller\_id),

FOREIGN KEY (seller\_id) REFERENCES SELLER(seller\_id)

ON DELETE CASCADE

);

CREATE TABLE HISTORY (

history\_id INT,

buyer\_id INT NOT NULL,

date DATETIME DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (history\_id, buyer\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE QUESTION (

question\_id INT NOT NULL AUTO\_INCREMENT,

buyer\_id INT NOT NULL,

question\_txt TEXT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (question\_id, buyer\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE ANSWER (

answer\_id INT NOT NULL AUTO\_INCREMENT,

seller\_id INT NOT NULL,

buyer\_id INT NOT NULL,

answer\_txt TEXT NOT NULL,

question\_id INT NOT NULL,

PRIMARY KEY (answer\_id, seller\_id),

FOREIGN KEY (question\_id) REFERENCES QUESTION(question\_id)

ON DELETE CASCADE,

FOREIGN KEY (seller\_id) REFERENCES SELLER(seller\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES QUESTION(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE APPLIES\_STORE\_DISCOUNT (

seller\_id INT NOT NULL,

store\_id INT NOT NULL,

discount\_id INT NOT NULL,

PRIMARY KEY (seller\_id, store\_id, discount\_id),

FOREIGN KEY (seller\_id) REFERENCES STORE(seller\_id)

ON DELETE CASCADE,

FOREIGN KEY (store\_id) REFERENCES STORE(store\_id)

ON DELETE CASCADE,

FOREIGN KEY (discount\_id) REFERENCES DISCOUNT(discount\_id)

ON DELETE CASCADE

);

CREATE TABLE FOLLOWS\_TABLE (

buyer\_id INT NOT NULL,

store\_id INT NOT NULL,

seller\_id INT NOT NULL,

date DATETIME DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (buyer\_id, store\_id, seller\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (store\_id) REFERENCES STORE(store\_id)

ON DELETE CASCADE,

FOREIGN KEY (seller\_id) REFERENCES STORE(seller\_id)

ON DELETE CASCADE

);

CREATE TABLE FAVORITES (

buyer\_id INT NOT NULL,

product\_id INT NOT NULL,

added\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

is\_active BOOLEAN DEFAULT TRUE,

PRIMARY KEY (buyer\_id, product\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE LISTS (

buyer\_id INT NOT NULL,

product\_id INT NOT NULL,

list\_name VARCHAR(255),

created\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (buyer\_id, product\_id),

FOREIGN KEY (buyer\_id) REFERENCES BUYER(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE SHOWS (

buyer\_id INT NOT NULL,

history\_id INT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (buyer\_id, history\_id, product\_id),

FOREIGN KEY (buyer\_id) REFERENCES HISTORY(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (history\_id) REFERENCES HISTORY(history\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE APPLIES\_CART\_DISCOUNT (

discount\_id INT NOT NULL,

cart\_id INT NOT NULL,

buyer\_id INT NOT NULL,

PRIMARY KEY (discount\_id, cart\_id, buyer\_id),

FOREIGN KEY (discount\_id) REFERENCES DISCOUNT(discount\_id)

ON DELETE CASCADE,

FOREIGN KEY (cart\_id) REFERENCES CART(cart\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES CART(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE SELLS (

store\_id INT NOT NULL,

seller\_id INT NOT NULL,

product\_id INT NOT NULL,

PRIMARY KEY (store\_id, seller\_id, product\_id),

FOREIGN KEY (store\_id) REFERENCES STORE(store\_id)

ON DELETE CASCADE,

FOREIGN KEY (seller\_id) REFERENCES STORE(seller\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE

);

CREATE TABLE SHIPMENT (

shipment\_id INT NOT NULL,

order\_id INT NOT NULL,

buyer\_id INT NOT NULL,

status VARCHAR(255),

country VARCHAR(255),

shipping\_date DATETIME,

expected\_date DATETIME,

arrival\_date DATETIME,

flat\_rate DECIMAL(10,2) DEFAULT 0.00,

extra\_cost DECIMAL(10,2) DEFAULT 0.00,

shipment\_type VARCHAR(50),

PRIMARY KEY (shipment\_id, order\_id, buyer\_id),

FOREIGN KEY (order\_id) REFERENCES ORDERTABLE(order\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES ORDERTABLE(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE ORDER\_HISTORY (

order\_h\_id INT NOT NULL,

order\_id INT NOT NULL,

buyer\_id INT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

event\_type VARCHAR(255),

tracking\_no VARCHAR(255),

PRIMARY KEY (order\_h\_id, order\_id, buyer\_id),

FOREIGN KEY (order\_id) REFERENCES ORDERTABLE(order\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES ORDERTABLE(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE SHIPPED\_BY (

shipment\_id INT NOT NULL,

order\_id INT NOT NULL,

buyer\_id INT NOT NULL,

carrier\_id INT NOT NULL,

PRIMARY KEY (shipment\_id, order\_id, buyer\_id, carrier\_id),

FOREIGN KEY (shipment\_id) REFERENCES SHIPMENT(shipment\_id)

ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES SHIPMENT(order\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES SHIPMENT(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (carrier\_id) REFERENCES CARRIER(carrier\_id)

ON DELETE CASCADE

);

CREATE TABLE ORDER\_EVENT (

event\_id INT NOT NULL,

order\_h\_id INT NOT NULL,

order\_id INT NOT NULL,

buyer\_id INT NOT NULL,

name VARCHAR(255),

update\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

reason\_txt TEXT,

type VARCHAR(50),

PRIMARY KEY (event\_id, order\_h\_id, order\_id, buyer\_id),

FOREIGN KEY (order\_h\_id) REFERENCES ORDER\_HISTORY(order\_h\_id)

ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES ORDER\_HISTORY(order\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES ORDER\_HISTORY(buyer\_id)

ON DELETE CASCADE

);

CREATE TABLE TRACKS\_PRODUCT\_ORDER\_EVENT (

product\_id INT NOT NULL,

order\_id INT NOT NULL,

buyer\_id INT NOT NULL,

event\_id INT NOT NULL,

order\_h\_id INT NOT NULL,

PRIMARY KEY (product\_id, order\_id, buyer\_id, event\_id, order\_h\_id),

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE,

FOREIGN KEY (order\_id) REFERENCES ORDER\_EVENT(order\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES ORDER\_EVENT(buyer\_id)

ON DELETE CASCADE,

FOREIGN KEY (event\_id) REFERENCES ORDER\_EVENT(event\_id)

ON DELETE CASCADE,

FOREIGN KEY (order\_h\_id) REFERENCES ORDER\_EVENT(order\_h\_id)

ON DELETE CASCADE

);

CREATE TABLE CART\_ITEMS (

cart\_item\_id INT NOT NULL,

cart\_id INT NOT NULL,

product\_id INT NOT NULL,

buyer\_id INT NOT NULL,

quantity INT DEFAULT 1,

price DECIMAL(10, 2) NOT NULL,

PRIMARY KEY (cart\_item\_id, product\_id),

FOREIGN KEY (cart\_id) REFERENCES CART(cart\_id)

ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES PRODUCT(product\_id)

ON DELETE CASCADE,

FOREIGN KEY (buyer\_id) REFERENCES CART(buyer\_id)

ON DELETE CASCADE

);

--------------------------------------------------------------------------------

-- TRIGGERS

--------------------------------------------------------------------------------

DELIMITER $$

-- 1) Before inserting a CART\_ITEM, ensure CART exists for that buyer; if not, create it

CREATE TRIGGER before\_cart\_item\_insert

BEFORE INSERT ON CART\_ITEMS

FOR EACH ROW

BEGIN

DECLARE cartExists INT DEFAULT 0;

-- We assume NEW.cart\_id actually stores "buyer\_id" in your original logic,

-- which is why you're checking CART.buyer\_id = NEW.cart\_id.

-- Adjust if needed (common usage is you store an actual cart\_id).

SELECT COUNT(\*)

INTO cartExists

FROM CART

WHERE buyer\_id = NEW.cart\_id;

IF cartExists = 0 THEN

INSERT INTO CART (buyer\_id) VALUES (NEW.cart\_id);

SET NEW.cart\_id = LAST\_INSERT\_ID();

END IF;

END$$

DELIMITER ;

DELIMITER $$

-- 2) After inserting an order, auto-insert shipping details

CREATE TRIGGER after\_order\_insert

AFTER INSERT ON ORDERTABLE

FOR EACH ROW

BEGIN

INSERT INTO SHIPPING\_DETAILS (

buyer\_id, order\_id, phone, city, state, zip, country, street\_address

)

VALUES (

NEW.buyer\_id, NEW.order\_id,

'05000000000', 'default\_city', 'default\_state', '0000', 'default\_country', 'default\_street'

);

END$$

DELIMITER ;

DROP TRIGGER IF EXISTS before\_buyer\_insert$$

DELIMITER $$

CREATE TRIGGER before\_buyer\_insert

BEFORE INSERT ON BUYER

FOR EACH ROW

BEGIN

INSERT INTO USERTABLE (

user\_id, e\_mail, phone, password, reg\_date, lname, fname

)

VALUES (

NEW.buyer\_id,

CONCAT('user', NEW.buyer\_id, '@example.com'),

CONCAT('555000', LPAD(NEW.buyer\_id, 4, '0')), -- Örneğin: 5550000001

'defaultpassword',

NOW(),

'Default',

'User'

);

-- Aynı anda registered user kaydı oluşturuluyor

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (NEW.buyer\_id);

END$$

DELIMITER ;

-- TRIGGER FINISHED

-- Seller 1 için kayıtlar

INSERT INTO USERTABLE (user\_id, e\_mail, phone, password, reg\_date, lname, fname)

VALUES (10, 'alice@example.com', '1111111111', 'alicepass', NOW(), 'Smith', 'Alice');

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (10);

INSERT INTO SELLER (seller\_id)

VALUES (10);

-- Seller 2 için kayıtlar

INSERT INTO USERTABLE (user\_id, e\_mail, phone, password, reg\_date, lname, fname)

VALUES (11, 'bob@example.com', '2222222222', 'bobpass', NOW(), 'Johnson', 'Bob');

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (11);

INSERT INTO SELLER (seller\_id)

VALUES (11);

-- Seller 3 için kayıtlar

INSERT INTO USERTABLE (user\_id, e\_mail, phone, password, reg\_date, lname, fname)

VALUES (12, 'carol@example.com', '3333333333', 'carolpass', NOW(), 'Williams', 'Carol');

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (12);

INSERT INTO SELLER (seller\_id)

VALUES (12);

-- Seller 4 için kayıtlar

INSERT INTO USERTABLE (user\_id, e\_mail, phone, password, reg\_date, lname, fname)

VALUES (13, 'dave@example.com', '4444444444', 'davepass', NOW(), 'Brown', 'Dave');

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (13);

INSERT INTO SELLER (seller\_id)

VALUES (13);

-- Seller 5 için kayıtlar

INSERT INTO USERTABLE (user\_id, e\_mail, phone, password, reg\_date, lname, fname)

VALUES (14, 'eve@example.com', '5555555555', 'evepass', NOW(), 'Davis', 'Eve');

INSERT INTO REGISTERED\_USER (user\_id)

VALUES (14);

INSERT INTO SELLER (seller\_id)

VALUES (14);

-- Insert Dıscounts

-- Discount 1

INSERT INTO DISCOUNT (

min\_amount,

type,

valid\_from,

expiry\_date,

amount,

coupon\_code,

is\_standard,

is\_coupon

) VALUES (

100.00,

'Store',

CURDATE(),

DATE\_ADD(CURDATE(), INTERVAL 1 MONTH),

10.00,

'WINTER10',

TRUE,

FALSE

);

-- Discount 2

INSERT INTO DISCOUNT (

min\_amount,

type,

valid\_from,

expiry\_date,

amount,

coupon\_code,

is\_standard,

is\_coupon

) VALUES (

100.00,

'Store',

CURDATE(),

DATE\_ADD(CURDATE(), INTERVAL 1 MONTH),

10.00,

'WINTER10',

TRUE,

FALSE

);

-- Discount 3

INSERT INTO DISCOUNT (

min\_amount,

type,

valid\_from,

expiry\_date,

amount,

coupon\_code,

is\_standard,

is\_coupon

) VALUES (

100.00,

'Store',

CURDATE(),

DATE\_ADD(CURDATE(), INTERVAL 1 MONTH),

10.00,

'WINTER10',

TRUE,

FALSE

);

INSERT INTO STORE (seller\_id, store\_name)

VALUES (10, 'A Boutique');

INSERT INTO STORE (seller\_id, store\_name)

VALUES (11, 'B Boutique');

INSERT INTO STORE (seller\_id, store\_name)

VALUES (12, 'C Boutique');

INSERT INTO APPLIES\_STORE\_DISCOUNT (seller\_id, store\_id, discount\_id)

VALUES (10, 1, 1);

INSERT INTO APPLIES\_STORE\_DISCOUNT (seller\_id, store\_id, discount\_id)

VALUES (10, 2, 2);

-- -----------------------------

-- BUYER INSERTS

-- --------------------------------

-- 1. Buyer

INSERT INTO BUYER (buyer\_id, start\_date, end\_date, is\_standard\_member, is\_premium\_member)

VALUES (1, '2024-01-01', '2024-12-31', TRUE, FALSE);

UPDATE USERTABLE

SET

e\_mail = 'burak@example.com',

phone = '5351234567',

password = '12345',

lname = 'taskın',

fname = 'Burak'

WHERE user\_id = 1;

-- 2. Buyer ekleme: is\_standard\_member = TRUE

INSERT INTO BUYER (buyer\_id, start\_date, end\_date, is\_standard\_member, is\_premium\_member)

VALUES (2, '2024-02-01', '2024-11-30', TRUE, FALSE);

UPDATE USERTABLE

SET

e\_mail = 'ismail@example.com',

phone = '52531234567',

password = '123xvdf45',

lname = 'Erkan',

fname = 'İsmail'

WHERE user\_id = 2;

INSERT INTO BUYER (buyer\_id, start\_date, end\_date, is\_standard\_member, is\_premium\_member)

VALUES (3, '2024-03-01', NULL, FALSE, TRUE);

UPDATE USERTABLE

SET

e\_mail = 'tuna@example.com',

phone = '53531234567',

password = '122dca345',

lname = 'Baskurt',

fname = 'Tuna'

WHERE user\_id = 3;

-- 4. Buyer ekleme: is\_standard\_member = TRUE

INSERT INTO BUYER (buyer\_id, start\_date, end\_date, is\_standard\_member, is\_premium\_member)

VALUES (4, '2024-04-01', '2025-03-31', TRUE, FALSE);

UPDATE USERTABLE

SET

e\_mail = 'iremnur@example.com',

phone = '5442234567',

password = '122dca345',

lname = 'İremnur',

fname = 'Sener'

WHERE user\_id = 4;

-- 5. Buyer ekleme: is\_standard\_member = FALSE

INSERT INTO BUYER (buyer\_id, start\_date, end\_date, is\_standard\_member, is\_premium\_member)

VALUES (5, '2024-05-01', NULL, FALSE, TRUE);

UPDATE USERTABLE

SET

e\_mail = 'merve@example.com',

phone = '53251234567',

password = '12ad22dca345',

lname = 'Merve',

fname = 'Taşkın'

WHERE user\_id = 5;

-- -----------------------------------

-- ------------------ 1. Insert into PAYMENT\_METHOD

INSERT INTO PAYMENT\_METHOD (

buyer\_id, payment\_type, holder\_name, isDefault, exp\_date, cvv, google\_mail, current\_balance

) VALUES

(1, 'Credit Card', 'Burak Taskın', TRUE, '2025-12-31', '123', 'burak@example.com', 450.00),

(2, 'PayPal', 'İsmail Erkan', TRUE, '2025-11-30', '456', 'ismail@example.com', 300.00),

(3, 'Debit Card', 'Tuna Baskurt', TRUE, '2026-01-15', '789', 'tuna@example.com', 400.00),

(4, 'Credit Card', 'İremnur Sener', TRUE, '2025-03-31', '012', 'iremnur@example.com', 600.00),

(5, 'PayPal', 'Merve Taşkın', TRUE, '2025-05-31', '345', 'merve@example.com', 200.00);

-- Insert into CART

INSERT INTO CART (buyer\_id)

VALUES

(1),

(2),

(3),

(4),

(5);

-- Insert into ORDERTABLE with total > 100

INSERT INTO ORDERTABLE (

buyer\_id, payment\_id, total, cart\_id

) VALUES

(1, 1, 150.00, 1),

(2, 2, 120.00, 2),

(3, 3, 90.00, 3),

(4, 4, 120.00, 4),

(5, 5, 130.00, 5);

-- Update SHIPPING\_DETAILS with actual shipping information

UPDATE SHIPPING\_DETAILS

SET

phone = CASE buyer\_id

WHEN 1 THEN '5351234567'

WHEN 2 THEN '5253123456'

WHEN 3 THEN '5353123456'

WHEN 4 THEN '5442234567'

WHEN 5 THEN '5325123456'

END,

city = CASE buyer\_id

WHEN 1 THEN 'Istanbul'

WHEN 2 THEN 'Ankara'

WHEN 3 THEN 'Izmir'

WHEN 4 THEN 'Bursa'

WHEN 5 THEN 'Antalya'

END,

state = CASE buyer\_id

WHEN 1 THEN 'kadıkoy'

WHEN 2 THEN 'mamak'

WHEN 3 THEN 'konak'

WHEN 4 THEN 'mudanya'

WHEN 5 THEN 'kaş'

END,

zip = CASE buyer\_id

WHEN 1 THEN '34000'

WHEN 2 THEN '06000'

WHEN 3 THEN '35000'

WHEN 4 THEN '16000'

WHEN 5 THEN '07000'

END,

country = 'Turkey',

street\_address = CASE buyer\_id

WHEN 1 THEN '123 St'

WHEN 2 THEN '456 St'

WHEN 3 THEN '789 St'

WHEN 4 THEN '101 St'

WHEN 5 THEN '202 St'

END

WHERE order\_id IN (1, 2, 3, 4, 5);

-- ------------ ------------ ------------ ------------ ------------

INSERT INTO BRAND (brand\_id, name) VALUES

(1, 'Lufian'),

(2, 'Zara'),

(3, 'Mango'),

(4, 'Bershka'),

(5, 'Pull&Bear');

INSERT INTO PRODUCT (brand\_id, review\_count, name, is\_flash, descp, stock\_level, is\_best\_seller, price, size)

VALUES

-- Lufian Products

(1, 120, 'Lufian Classic Shirt', FALSE, 'Elegant classic shirt for all occasions', 150, TRUE, 59.99, 'M'),

(1, 75, 'Lufian Casual Pants', TRUE, 'Comfortable casual pants with a modern fit', 80, FALSE, 39.99, 'L'),

-- Zara Products

(2, 200, 'Zara Summer Dress', FALSE, 'Lightweight summer dress with floral patterns', 200, TRUE, 89.99, 'S'),

-- (2, 150, 'Zara Denim Jacket', TRUE, 'Stylish denim jacket perfect for any season', 120, TRUE, 129.99, 'M'),

-- Mango Products

(3, 180, 'Mango Knit Sweater', FALSE, 'Cozy knit sweater for chilly days', 100, TRUE, 49.99, 'L'),

(3, 95, 'Mango Leather Belt', TRUE, 'Premium leather belt with sleek design', 60, FALSE, 29.99, 'One Size'),

-- Bershka Products

-- (4, 130, 'Bershka Sport T-Shirt', FALSE, 'Breathable sport t-shirt for active lifestyles', 140, TRUE, 24.99, 'M'),

(4, 85, 'Bershka Trendy Shorts', TRUE, 'Fashionable shorts with a comfortable fit', 90, FALSE, 19.99, 'L'),

-- Pull&Bear Products

(5, 160, 'Pull&Bear Graphic Hoodie', FALSE, 'Stylish hoodie with unique graphic prints', 110, TRUE, 54.99, 'XL'),

(5, 100, 'Pull&Bear Chino Shorts', TRUE, 'Classic chino shorts for everyday wear', 70, FALSE, 34.99, 'M');

INSERT INTO CATEGORY (name) VALUES

('shirt'), -- category\_id = 1

('pants'), -- category\_id = 2

('dress'), -- category\_id = 3

('sweater'), -- category\_id = 4

('belt'), -- category\_id = 5

('shorts'), -- category\_id = 6

('hoodie'); -- category\_id = 7

INSERT INTO CART\_ITEMS (cart\_item\_id, cart\_id, product\_id, buyer\_id, quantity, price) VALUES

-- Cart 1 Items (buyer\_id 1)

(1, 1, 1, 1, 2, 59.99),

(2, 1, 2, 1, 1, 39.99),

-- Cart 2 Items (buyer\_id 2)

(1, 2, 3, 2, 1, 89.99),

(2, 2, 4, 2, 2, 129.99),

-- Cart 3 Items (buyer\_id 3)

(1, 3, 5, 3, 1, 29.99),

(2, 3, 3, 3, 2, 24.99),

-- Cart 4 Items (buyer\_id 4)

(2, 4, 1, 4, 1, 19.99),

(2, 4, 6, 4, 3, 54.99),

-- Cart 5 Items (buyer\_id 5)

(1, 5, 2, 5, 1, 34.99),

(2, 5, 5, 5, 2, 49.99);

INSERT INTO REVIEW (title, rating, buyer\_id, is\_order\_delivered)

VALUES

('Great product', 5, 1, TRUE),

('Good service', 4, 2, TRUE),

('Excellent quality', 5, 3, TRUE),

('bad experience', 1, 4, TRUE),

('Very useful', 5, 5, TRUE);

INSERT INTO PRODUCT\_REVIEW (review\_id, product\_id)

VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5);

SET SQL\_SAFE\_UPDATES = 0;

-- UPDATE

-- Increase the price of all products of the “Lufian” brand by 210%

UPDATE PRODUCT AS p

JOIN BRAND AS b ON p.brand\_id = b.brand\_id

SET p.price = p.price \* 3.10

WHERE b.name = 'Lufian';

-- select \* from product;

-- update city and state

UPDATE SHIPPING\_DETAILS AS sd

JOIN ORDERTABLE AS o

ON sd.order\_id = o.order\_id

AND sd.buyer\_id = o.buyer\_id

JOIN USERTABLE AS u

ON o.buyer\_id = u.user\_id

SET sd.city = 'Mardin',

sd.state = 'Kızıltepe'

WHERE u.e\_mail = 'burak@example.com';

-- select \* from SHIPPING\_DETAILS;

-- E-postası “alice@example.com” olan satıcının dükkan adının sonuna “ - VIP” eklemek istiyoruz.

UPDATE STORE AS st

JOIN SELLER AS se ON st.seller\_id = se.seller\_id

JOIN USERTABLE AS u ON se.seller\_id = u.user\_id

SET st.store\_name = CONCAT(st.store\_name, ' - VIP')

WHERE u.e\_mail = 'alice@example.com';

select \* from usertable;

-- --------------- -----------------------------------------------------------

-- Tüm Siparişlerin ve İlgili Teslimat Detaylarının Listelenmesi

--------------------------------------------------------------------------------

-- EXAMPLE UPDATE QUERIES

--------------------------------------------------------------------------------

/\*

UPDATE USERTABLE

SET e\_mail = 'new23232email@example.com'

WHERE user\_id = 1;

\*/

/\*

UPDATE PRODUCT

SET price = 899.99

WHERE product\_id = 1;

\*/

/\*

UPDATE ORDERTABLE

SET total = 150.75

WHERE order\_id = 1;

\*/

/\*

UPDATE ORDERTABLE

SET payment\_id = 2

WHERE order\_id = 4;

\*/

--------------------------------------------------------------------------------

-- EXAMPLE DELETE QUERIES

--------------------------------------------------------------------------------

/\*

DELETE FROM PRODUCT

WHERE product\_id = 1;

\*/

/\*

SET SQL\_SAFE\_UPDATES = 0;

-- Bir markaya ait tüm ürünleri ve bu ürünlerle ilişkili tüm kayıtları (örneğin, sepet ve inceleme bilgileri) silmek :

DELETE ci, pr, p, b

FROM BRAND b

LEFT JOIN PRODUCT p ON b.brand\_id = p.brand\_id

LEFT JOIN CART\_ITEMS ci ON p.product\_id = ci.product\_id

LEFT JOIN PRODUCT\_REVIEW pr ON p.product\_id = pr.product\_id

WHERE b.name = 'Lufian';

\*/

-- Belirli Bir Kullanıcıya Ait Tüm Siparişleri ve Teslimat Bilgilerini Silme

/\*

DELETE o, s

FROM ORDERTABLE o

LEFT JOIN SHIPPING\_DETAILS s ON o.order\_id = s.order\_id

WHERE o.buyer\_id = 1;

\*/

--------------------------------------------------------------------------------

-- EXAMPLE SELECT QUERIES

--------------------------------------------------------------------------------

/\*

SELECT user\_id, e\_mail, phone, fname, lname, reg\_date

FROM USERTABLE ;

\*/

/\*

-- Tüm Alıcıların E-posta ve Telefon Bilgilerini Listeleme

SELECT

b.buyer\_id,

u.e\_mail,

u.phone,

u.fname,

u.lname

FROM

BUYER b

JOIN

USERTABLE u ON b.buyer\_id = u.user\_id;

\*/

-- Tüm Siparişlerin ve İlgili Teslimat Detaylarının Listelenmesi

/\*

SELECT

o.order\_id,

o.total,

o.date AS order\_date,

s.city,

s.state,

s.zip,

s.country,

s.street\_address

FROM

ORDERTABLE o

JOIN

SHIPPING\_DETAILS s ON o.order\_id = s.order\_id;

\*/

-- Favori Ürünleri Olan Alıcılar ve Ürün Detayları

/\*

SELECT

b.buyer\_id,

u.e\_mail,

p.product\_id,

p.name AS product\_name,

p.price

FROM

BUYER b

JOIN

USERTABLE u ON b.buyer\_id = u.user\_id

JOIN

FAVORITES f ON b.buyer\_id = f.buyer\_id

JOIN

PRODUCT p ON f.product\_id = p.product\_id;

\*/

-- Yüksek Bakiyeye Sahip Alıcılar ve Ödeme Yöntemleri

/\*

SELECT

b.buyer\_id,

u.e\_mail,

pm.payment\_type,

pm.current\_balance

FROM

BUYER b

JOIN

USERTABLE u ON b.buyer\_id = u.user\_id

JOIN

PAYMENT\_METHOD pm ON b.buyer\_id = pm.buyer\_id

WHERE

pm.current\_balance > 100;

\*/

---------------------------

-- 5 CRITICAL SELECT STATEMENT

-------------------------------------------------

/\*

-- Example: List standard members

SELECT b.buyer\_id, u.e\_mail, u.phone

FROM BUYER b

JOIN USERTABLE u ON b.buyer\_id = u.user\_id

WHERE b.is\_standard\_member = TRUE;

\*/

-- Example: Orders with total > 100

/\*

SELECT o.order\_id, o.total, c.buyer\_id

FROM ORDERTABLE o

JOIN CART c ON o.cart\_id = c.cart\_id

WHERE o.total > 100;

\*/

-- Example: Best-seller products in orders

/\*

SELECT o.order\_id, p.name AS product\_name, o.total

FROM ORDERTABLE o

JOIN CART c ON o.cart\_id = c.cart\_id

JOIN CART\_ITEMS ci ON c.cart\_id = ci.cart\_id

JOIN PRODUCT p ON ci.product\_id = p.product\_id

WHERE p.is\_best\_seller = TRUE;

\*/

-- Example: Buyers with current\_balance > 100

/\*

SELECT b.buyer\_id, u.e\_mail, pm.payment\_type, pm.current\_balance

FROM BUYER b

JOIN USERTABLE u ON b.buyer\_id = u.user\_id

JOIN PAYMENT\_METHOD pm ON b.buyer\_id = pm.buyer\_id

WHERE pm.current\_balance > 100;

\*/

-- Example: High-rated reviews

/\*

SELECT r.review\_id, r.title, r.rating, u.e\_mail

FROM REVIEW r

JOIN BUYER b ON r.buyer\_id = b.buyer\_id

JOIN USERTABLE u ON b.buyer\_id = u.user\_id

WHERE r.rating >= 4;

\*/

-- Example: Show top review\_count products

/\*

SELECT p.product\_id, p.name, p.review\_count, b.name AS brand\_name

FROM PRODUCT p

JOIN BRAND b ON p.brand\_id = b.brand\_id

ORDER BY p.review\_count DESC;

\*/

-- Example: Product categories and how many times they are ordered

/\*

SELECT c.name AS category\_name, COUNT(ci.product\_id) AS total\_products\_ordered

FROM CART\_ITEMS ci

JOIN PRODUCT p ON ci.product\_id = p.product\_id

JOIN CATEGORY c ON p.brand\_id = c.category\_id -- This line is just an example join (adjust as needed)

GROUP BY c.name

ORDER BY total\_products\_ordered DESC;

\*/