```sql
CREATE TABLE SHOP (

   shopID INT PRIMARY KEY,

   name VARCHAR(255) NOT NULL,

   contact VARCHAR(100),

   location VARCHAR(255)

);

CREATE TABLE CUSTOMER (

   customerID INT PRIMARY KEY,

   email VARCHAR(255) NOT NULL UNIQUE,

   passwd VARCHAR(255) NOT NULL,

   total_point INT,

   isElite BOOLEAN,

   phonenum VARCHAR(15),

   address TEXT,

   gender VARCHAR(10),

   name VARCHAR(100),

   surname VARCHAR(100),

   bday DATE,

   weight FLOAT,

   height FLOAT

);

CREATE TABLE PAYMENT_METHOD (

   paymentID INT,

   customerID INT,

   currency VARCHAR(50),

   PRIMARY KEY (customerID, paymentID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE PRODUCT (
    productID INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT
);

CREATE TABLE CATEGORY (
    category_name VARCHAR(100) PRIMARY KEY
);

CREATE TABLE LOGISTICS_COMPANY (
    company_name VARCHAR(255) PRIMARY KEY
);

CREATE TABLE GIFTCARD (
    customerID INT,
    paymentID INT,
    card_code VARCHAR(50),
    balance FLOAT,
    PRIMARY KEY (customerID, paymentID, card_code),
    FOREIGN KEY (customerID, paymentID) REFERENCES
PAYMENT_METHOD(customerID, paymentID)
);

CREATE TABLE WISHLIST (
    customerID INT,
    listID INT,
    name VARCHAR(255),
    num_products INT,
    PRIMARY KEY (customerID, listID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE FAVOURITE (
        customerID INT,
    listID INT,
    name VARCHAR(255) DEFAULT 'FAVOURITES',
    num_products INT,
        PRIMARY KEY (customerID, listID),
    FOREIGN KEY (customerID, listID) REFERENCES WISHLIST(customerID, listID)
);

CREATE TABLE SHARED_WISHLIST (
    customerID INT,
    listID INT,
    num_views INT,
    PRIMARY KEY (customerID, listID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE PRIVATE_WISHLIST (
    customerID INT,
    listID INT,
    PRIMARY KEY (customerID, listID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE PUBLIC_WISHLIST (
    customerID INT,
    listID INT,
    num_views INT,
    num_followers INT,
    public_type VARCHAR(50),
    PRIMARY KEY (customerID, listID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE `ORDER` (
    customerID INT,
    orderID INT,
    status VARCHAR(50),
    final_price FLOAT,
    PRIMARY KEY (customerID, orderID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE CART (
    customerID INT,
    total_price FLOAT,
    discount FLOAT,
    num_products INT,
    PRIMARY KEY (customerID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE SHOP_PRODUCT (
    shopID INT,
    productID INT,
    stock_level INT,
    price FLOAT,
    PRIMARY KEY (shopID, productID),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (productID) REFERENCES PRODUCT(productID)
);



CREATE TABLE PRODUCT_REVIEW (
        comment TEXT,
    rating INT,
    productID INT,
    customerID INT,
    PRIMARY KEY (productID, customerID),
    FOREIGN KEY (productID) REFERENCES PRODUCT(productID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE SHOPIER_REVIEW (
    comment TEXT,
    rating INT,
    shopID INT,
    customerID INT,
    PRIMARY KEY (shopID, customerID),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE LOGISTICS (
    company_name VARCHAR(255),
    shopID INT,
    customerID INT,
    orderID INT,
    logisticsID INT,
    phonenum VARCHAR(15),
    delivery_address TEXT,
    receiver_name VARCHAR(255),
    delivery_date DATE,
    PRIMARY KEY (company_name, shopID, customerID, orderID, logisticsID),
    FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (customerID, orderID) references `ORDER` (customerID, orderID)
);

CREATE TABLE REFUND_TO (
    shopID INT,
    customerID INT,
    paymentID INT,
    money_amount FLOAT,
    PRIMARY KEY (shopID, customerID, paymentID),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (customerID, paymentID) REFERENCES
PAYMENT_METHOD(customerID, paymentID)
);

CREATE TABLE LIST_SPRODUCT (
    customerID INT,
    listID INT,
    shopID INT,
    productID INT,
    PRIMARY KEY (customerID, listID, shopID, productID),
    FOREIGN KEY (customerID, listID) REFERENCES WISHLIST(customerID, listID),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (productID) REFERENCES PRODUCT(productID)
);

CREATE TABLE CART_SPRODUCT (
    customerID INT,
    shopID INT,
    productID INT,
    PRIMARY KEY (customerID, shopID, productID),
    FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID),
    FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
    FOREIGN KEY (productID) REFERENCES PRODUCT(productID)
);
```

```
CREATE TABLE PRODUCT_CATEGORY (
  productID INT,
  category_name VARCHAR(100),
  PRIMARY KEY (productID, category_name),
  FOREIGN KEY (productID) REFERENCES PRODUCT(productID),
  FOREIGN KEY (category_name) REFERENCES CATEGORY(category_name)
);

CREATE TABLE WORKS_WITH (
  company_name VARCHAR(255),
  shopID INT,
  due_date DATE,
  PRIMARY KEY (company_name, shopID),
  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name),
  FOREIGN KEY (shopID) REFERENCES SHOP(shopID)
);

CREATE TABLE LIVE_VIDEO (
        customerID INT,
  customer_type varchar(15),
  start_time TIME,
  coupon_code VARCHAR(16)
);

CREATE TABLE PRODUCT_PHOTO (
  productID INT,
  photourl VARCHAR(255),
  PRIMARY KEY (productID, photourl),
  FOREIGN KEY (productID) REFERENCES PRODUCT(productID)
);

CREATE TABLE PRODUCT_REVIEW_PHOTO (
  productID INT,
  customerID INT,
  photourl VARCHAR(255),
  PRIMARY KEY (productID, customerID, photourl),
  FOREIGN KEY (productID) REFERENCES PRODUCT(productID),
  FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE GLOBAL_LOGISTICS_COMPANY_COUNTRIES (
  company_name VARCHAR(255),
  country VARCHAR(100),
  PRIMARY KEY (company_name, country),
  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name)
);

CREATE TABLE PACKET_SERVICE_LOGISTICS_COMPANY_TYPE (
  company_name VARCHAR(255),
  type VARCHAR(100),
  PRIMARY KEY (company_name, type),
  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name)
);
```

```
CREATE TABLE ORDER_SPRODUCT (
  customerID INT,
  orderID INT,
  shopID INT,
  productID INT,
  quantity INT,
  PRIMARY KEY (customerID, orderID, shopID, productID),
  FOREIGN KEY (customerID, orderID) REFERENCES `ORDER` (customerID, orderID),
  FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
  FOREIGN KEY (productID) REFERENCES PRODUCT(productID)
);

CREATE TABLE SHARED_WISHLIST_COLLABORATOR (
  customerID INT,
  listID INT,
  collaborator VARCHAR(255),
  PRIMARY KEY (customerID, listID, collaborator),
  FOREIGN KEY (customerID, listID) REFERENCES WISHLIST(customerID, listID)
);

CREATE TABLE INSTITUTIONAL_SHOP (
  shopID INT,
  PRIMARY KEY (shopID),
  FOREIGN KEY (shopID) REFERENCES SHOP(shopID)
);

CREATE TABLE INDEPENDENT_SHOP (
  shopID INT,
  customerID INT,
  PRIMARY KEY (shopID, customerID),
  FOREIGN KEY (shopID) REFERENCES SHOP(shopID),
  FOREIGN KEY (customerID) REFERENCES CUSTOMER(customerID)
);

CREATE TABLE CREDIT_CARD (
  customerID INT,
  paymentID INT,
  card_number VARCHAR(50),
  card_holder VARCHAR(255),
  cvv INT,
  PRIMARY KEY (customerID, paymentID),
  FOREIGN KEY (customerID, paymentID) REFERENCES
PAYMENT_METHOD(customerID, paymentID)
);

CREATE TABLE SHOPPING_CREDIT (
  customerID INT,
  paymentID INT,
  interest_rate FLOAT,
  type VARCHAR(50),
  loan_term INT,
  PRIMARY KEY (customerID, paymentID),
  FOREIGN KEY (customerID, paymentID) REFERENCES
PAYMENT_METHOD(customerID, paymentID)
);

CREATE TABLE HEPSI_PAY (
  customerID INT,

  paymentID INT,

  hepsipayID INT,

  balance FLOAT,

  PRIMARY KEY (customerID, paymentID),

  FOREIGN KEY (customerID, paymentID) REFERENCES
PAYMENT_METHOD(customerID, paymentID)

);


CREATE TABLE LOCAL_LOGISTICS_COMPANY (

  company_name VARCHAR(255),

  PRIMARY KEY (company_name),

  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name)

);


CREATE TABLE GLOBAL_LOGISTICS_COMPANY (

  company_name VARCHAR(255),

  duty FLOAT,

  PRIMARY KEY (company_name),

  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name)

);


CREATE TABLE PACKET_SERVICE_LOGISTICS_COMPANY (

  company_name VARCHAR(255),

  PRIMARY KEY (company_name),

  FOREIGN KEY (company_name) REFERENCES
LOGISTICS_COMPANY(company_name)
```

```
);

CREATE TABLE PAZARYERI (
   shopID INT,
   PRIMARY KEY (shopID),
   FOREIGN KEY (shopID) REFERENCES SHOP(shopID)
);

CREATE TABLE TRENDYOL_GO (
   shopID INT,
   min_price FLOAT,
   arrival_time INT,
   opening_hour TIME,
   closing_hour TIME,
   tgo_type VARCHAR(50),
   market_type VARCHAR(50),
   cuisine VARCHAR(50),
   PRIMARY KEY (shopID),
   FOREIGN KEY (shopID) REFERENCES SHOP(shopID)
);

-- total point update when product review done

DELIMITER //

CREATE TRIGGER update_total_point_after_product_review
AFTER INSERT ON PRODUCT_REVIEW
FOR EACH ROW
```

```
BEGIN
  DECLARE photo_count INT;
  SET photo_count = (SELECT COUNT(*)
            FROM PRODUCT_REVIEW_PHOTO
            WHERE productID = NEW.productID AND customerID = NEW.customerID);

  IF photo_count > 0 THEN
    UPDATE CUSTOMER
    SET total_point = total_point + 20
    WHERE customerID = NEW.customerID;
  ELSE
    UPDATE CUSTOMER
    SET total_point = total_point + 10
    WHERE customerID = NEW.customerID;
  END IF;
END;
//

DELIMITER ;

-- when total point is great or equal than 2500, isElite = true

DELIMITER //

CREATE TRIGGER check_isElite_before_total_point_update
BEFORE UPDATE ON CUSTOMER
FOR EACH ROW
BEGIN
```

```
  -- Eğer yeni toplam puan 2500 veya daha fazlaysa ve isElite henüz TRUE değilse
  IF NEW.total_point >= 2500 THEN
    SET NEW.isElite = TRUE; -- Yeni satırda isElite değerini güncelle
  END IF;
END;
//

DELIMITER ;

-- total point update when order done

DELIMITER //

CREATE TRIGGER add_points_after_order
AFTER INSERT ON `ORDER`
FOR EACH ROW
BEGIN
  DECLARE points_to_add INT;
  SET points_to_add = FLOOR(NEW.final_price / 150) * 15;

  IF points_to_add > 0 THEN
    UPDATE CUSTOMER
    SET total_point = total_point + points_to_add
    WHERE customerID = NEW.customerID;
  END IF;
END;
//
```

```
DELIMITER ;


-- update num_products on LIST_SPRODUCT when product added/removed
-- increment
DELIMITER //

CREATE TRIGGER increment_num_products_after_insert
AFTER INSERT ON LIST_SPRODUCT
FOR EACH ROW
BEGIN
   UPDATE WISHLIST
   SET num_products = num_products + 1
   WHERE customerID = NEW.customerID AND listID = NEW.listID;
END;
//

DELIMITER ;


-- decrement
DELIMITER //

CREATE TRIGGER decrement_num_products_after_delete
AFTER DELETE ON LIST_SPRODUCT
FOR EACH ROW
BEGIN
   UPDATE WISHLIST
   SET num_products = num_products - 1
   WHERE customerID = OLD.customerID AND listID = OLD.listID;
```

```sql
END;

//

DELIMITER ;

-- check constraints

ALTER TABLE CUSTOMER -- total point cant be negative

ADD CONSTRAINT chk_customer_total_point CHECK (total_point >= 0);

ALTER TABLE CUSTOMER -- customer must be one of those

ADD CONSTRAINT chk_customer_gender CHECK (gender IN ('Male', 'Female'));

ALTER TABLE CUSTOMER -- weight must be postiive

ADD CONSTRAINT chk_customer_weight CHECK (weight > 0);

ALTER TABLE CUSTOMER -- height must be positive

ADD CONSTRAINT chk_customer_height CHECK (height > 0);

ALTER TABLE PAYMENT_METHOD -- only three money currency is available

ADD CONSTRAINT chk_payment_currency CHECK (currency IN ('USD', 'EUR', 'TRY'));

ALTER TABLE REVIEW -- review must be between 1 and 5

ADD CONSTRAINT chk_review_rating_range CHECK (rating BETWEEN 1 AND 5);

ALTER TABLE SHOP_PRODUCT -- stock level cant be negative

ADD CONSTRAINT chk_shop_product_stock_level CHECK (stock_level >= 0);
```