EGE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT
INTRODUCTION TO DATABASES

DATABASE PROJECT

GROUP-38

PREPARED BY

05210000261-Bahrihan Torpil

05210000238-Mehmet Ali Avcı

05210000260-Kutlu Çağan Akın

ANALYSIS

1-AMAZON

In Amazon's e-commerce platform, users form the core of the system. Users are divided into two different roles: customers and sellers. A user can be both a customer and a seller or only take on one role. Users are identified with a unique ID in the platform. General information about users includes name, email, phone number, and password for security. Additionally, security verification options such as facial recognition and fingerprint features are available. Users can save multiple addresses, which can be used for delivery or billing purposes.

Customers are derived from the user entity and focus on purchasing transactions. Customers can be regular users or have a Prime membership option. Prime membership provides customers with faster delivery, special discounts, and additional benefits. Information related to the Prime membership, including subscription fees and renewal dates, is stored in the system. A customer uses a shopping cart to manage their purchases. Every customer has a cart, which remains active until the customer completes the purchase of the products. The products in the shopping cart are stored in the system along with quantity and price details. When a customer purchases the products in the cart, this information is converted into an order. Orders contain delivery address, payment method, and details of the ordered products. In connection with orders, invoices are generated for each customer on the Amazon platform. An invoice is linked to an order and contains the total amount of the order, taxes, and delivery fees.

Customers can store credit card information in the system to complete payment transactions more quickly and easily. Each stored card is identified with a unique ID and is associated with card number, expiration date, CVV, and cardholder name. This information allows customers to select cards they frequently use for payment transactions.

Customers can create lists to group and manage products they like. Each list is identified with a unique ID and is linked to a customer. This allows a customer to create multiple lists. The lists serve as structures where customers store their favorite products or products they plan to purchase later. For example, a customer can create lists like "My Favorites" or "Birthday Shopping." Each list can contain one or more products. Also, a product can appear in multiple lists.

Sellers are another entity derived from the user entity and are responsible for product management. Sellers can be individual or corporate, and their company name, location, and other details are stored in the system. On Amazon, the same product can be listed by different sellers. Sellers can set their own product prices,

which allows customers to compare prices. Additionally, sellers can apply percentage or fixed discounts to their products if they wish.

Products are organized within a broad category system, and each product can belong to a specific main category or subcategory. Along with products, images are also stored.

An order can contain multiple products, and these products may be shipped by different sellers. Amazon's logistics system is modeled to support this process. In the logistics system, each delivery is tracked with a tracking number and shipping date. The same order may contain products coming from different warehouses, and this requires each product to be included in separate logistics processes.

2-TRENDYOL

The system should show basic entities such as customers, sellers, products, orders, payments, logistics, and special offers, as well as the relationships between these entities in detail. At the core of the system are the customers. Each customer is identified by a unique ID number in the system, and multiple address details may be registered. Address information is used to determine the delivery points for customers to receive their orders.

Each customer has a personal account, and the security of this account is provided by the system. For security, password information is stored in the system, and two-factor authentication can be integrated when necessary.

Customers can be of two types: Elite and Standard. Customers earn points by making transactions on the platform and become Elite customers once they reach the required number of points.

Additionally, customers can register multiple credit cards in the system for making payments, and these cards can be used during the order process.

Customers can also organize their products into collections. Collections allow a customer to group products under specific themes or categories. The same product can appear in multiple collections.

Sellers are the key actors responsible for presenting products to customers. Each seller is identified by a unique ID. The seller's name, company name, location, and contact information are stored in the system.

Sellers display and manage their products on the platform. Each seller can add multiple products, which are associated with details such as price, stock status, brand, and description. Products are organized into specific categories and subcategories.

Sellers can also receive evaluations from customers. Customers can leave comments and ratings about a particular seller. These evaluations either increase or decrease the seller's reliability and play a crucial role in helping other customers make decisions. All evaluation and rating information is stored in the system.

In addition to managing their products, sellers can organize campaigns and special discounts on the Trendyol platform. These discounts can be applied on a product-by-product basis or can cover products in specific categories.

Products are among the core entities on the platform. Each product is identified by a unique ID in the system, and details such as the product name, brand, price, and stock status are stored. Product images are added to the system, making it easier for users to recognize the products. Products can also be evaluated by customers. Each evaluation is associated with a specific product and includes customer comments and ratings.

3-HEPSIBURADA

In Hepsiburada's e-commerce platform, users form the core of the system. Users are divided into two different roles: customers and sellers. A user can be both a customer and a seller or only take on one role. Users are identified with a unique ID in the platform. General information about users includes name, email, phone number, and password for security. Users can save multiple addresses, which can be used for delivery or billing purposes.

Customers are derived from the user entity and focus on purchasing transactions. A customer uses a shopping cart to manage their purchases. Each customer can have only one active shopping cart, and the cart's content, including product name, quantity, and price details, is stored in the system. Customers can also create lists to organize the products they like. Each list is associated with a customer, but a customer can have multiple lists, and the same product can appear in multiple lists. Customers have the ability to ask questions to sellers through the system. This feature creates a direct communication mechanism between the customer and the seller, allowing customers to obtain detailed information about products. Additionally, customers have invoices associated with their past orders, and each invoice contains the order total, taxes, and other additional charges.

Hepsiburada platform divides customers into two different types: standard and premium, based on their shopping activities and interactions with the platform. Standard customers can use the basic features of the platform, while premium customers benefit from broader advantages. Being a premium customer is associated with additional fees, and information such as subscription start date, renewal date, and the amount paid is stored in the system.

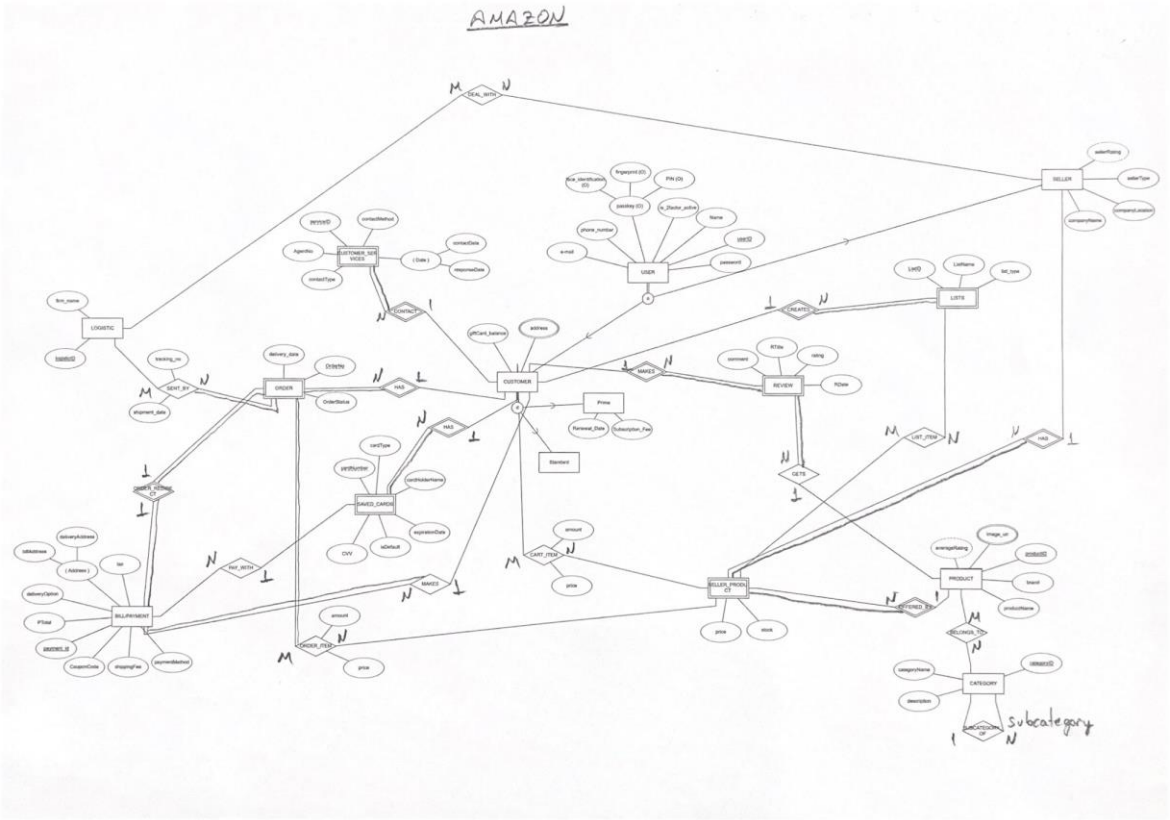
Hepsiburada platform offers special promotions such as campaigns and discounts to customers. Each special offer is identified by a unique ID in the system and can be associated with a specific product or category. Special offers provide price advantages for customers for a specific period. These offers are limited by a start date and end date. Each offer also has a name and description, which helps customers understand and evaluate the offer easily.

Sellers are another entity derived from the user entity and are responsible for presenting products on the platform. Sellers can list the same products at different prices. This allows customers to compare prices and creates a competitive shopping environment. Sellers can set product prices and apply percentage or fixed discounts to products if necessary. Sellers are also responsible for answering customer questions. This process is designed to improve customer satisfaction.

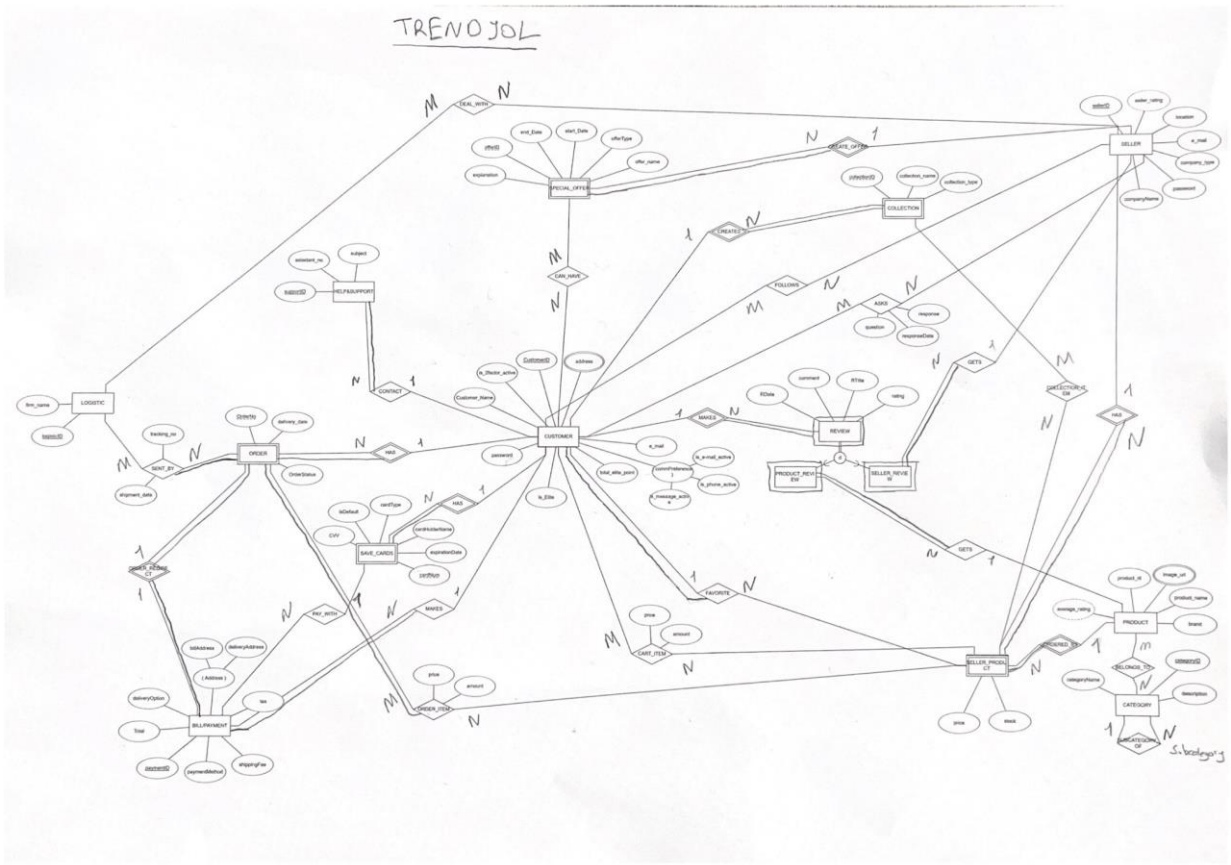
Products are organized within a broad category system, and each product is assigned to a specific main category or subcategory. For each product, unique details such as ID, name, price, brand, stock status, and description are stored. Product images are kept in the system for product promotion. The same product can be offered by different sellers at different prices. Additionally, some products are linked to user reviews and ratings. These reviews are used to evaluate product quality and share customer experiences.

DESIGN-CONCEPTUAL DESIGN

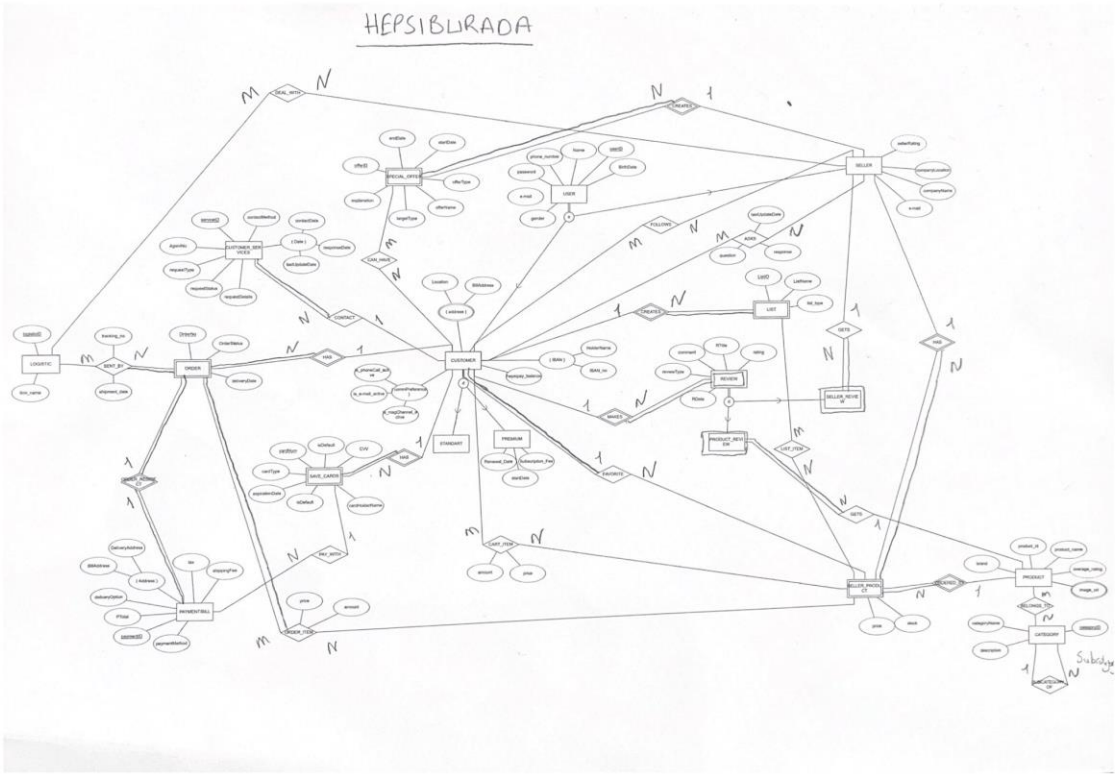
1-AMAZON DIAGRAM



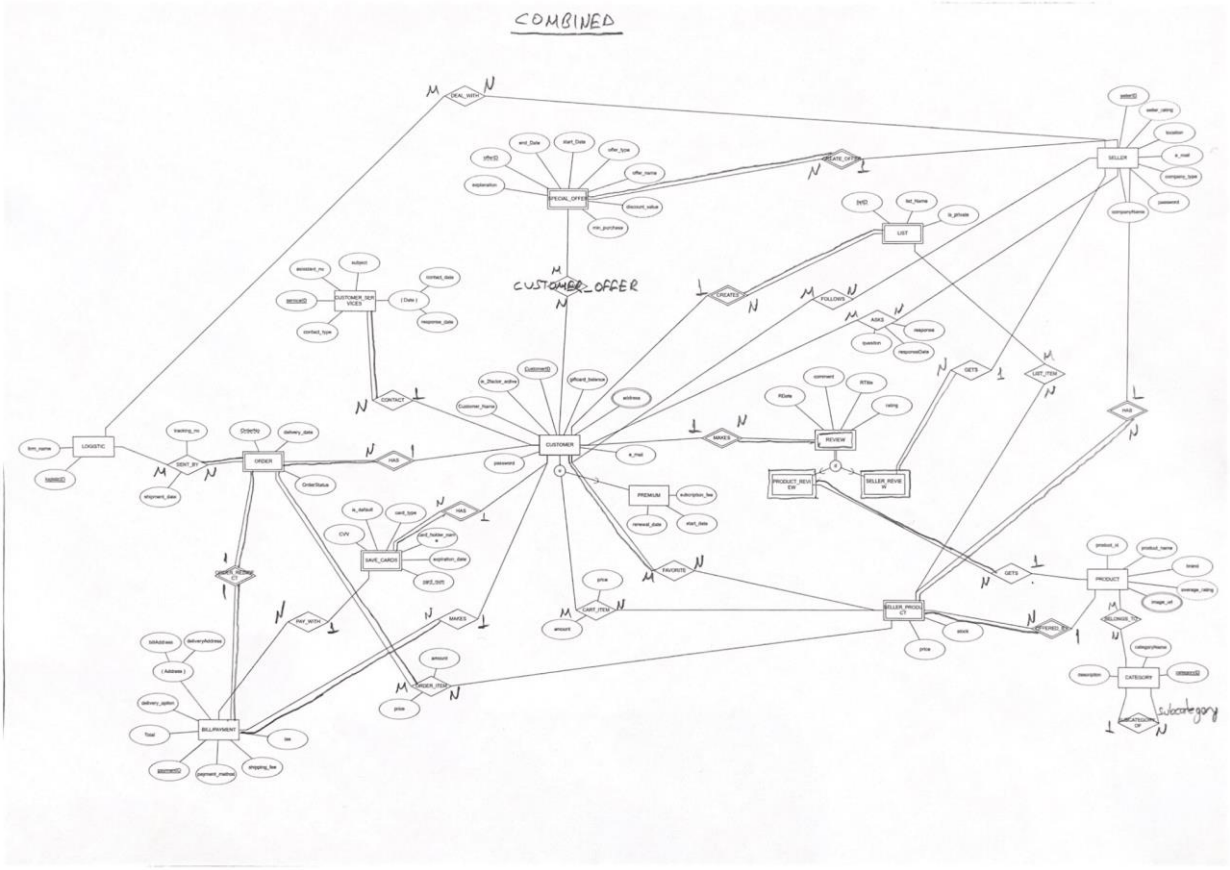
2-TRENDYOL DIAGRAM



3-HEPSIBURADA DIAGRAM



4-COMBINED DIAGRAM



How did we combine it?

We thoroughly analyzed the ER diagrams of Trendyol, Amazon, and Hepsiburada, considering the common features and differences among the three platforms to create a combined ER diagram. In this process, we integrated the unique features of each platform that could be harmonized with the others. Additionally, while identifying common entities and relationships, we paid close attention to avoiding redundant data by optimizing the use of entities and relationships.

DESIGN-LOGICAL MODEL

EER-to-RELATIONAL-MAPPING

1st ITERATION

STEP 1

CUSTOMER (customer_id, customer_name, e-mail, address, password, is_2factor_active, giftcard_balance, phone_number)

PRODUCT (product_id, product_name, brand, average_rating)

SELLER (seller_id, company_name, e-mail, password, location, company_type, seller_rating, phone_number)

BILL/PAYMENT (payment_id, payment_method, bill_address, delivery_address, delivery_option, total, tax, shipping_fee)

LOGISTIC (logistic_id, firm_name)

CUSTOMER_SERVICES (service_id, assistant_no, contact_date, response_date, contact_type)

CATEGORY (category_id, category_name)

Step 2

SAVED_CARD (card_num, CUSTOMER.customer_id, card_type, expiration_date, card_holder_name, CVV, is_default)

ORDER (order_no, CUSTOMER.customer_id, order_status, delivery_date)

REVIEW (CUSTOMER.customer_id, Rdate, Rtitle, comment, rating)

SELLER_PRODUCT(PRODUCT.product_id, SELLER.seller_id, stock, price)

PRODUCT_REVIEW (PRODUCT.product_id, Rdate, Rtitle, comment, rating)

SELLER_REVIEW (SELLER.seller_id, Rdate, Rtitle, comment, rating)

LIST (list_id, CUSTOMER.customer_id, list_name, list_type)

SPECIAL_OFFER (offer_id, SELLER.seller_id, offer_name, offer_type, start_date, end_date, explanation, discount_value, min_purchase)

Step 3

PAID_ORDERS (ORDER.order_no, BILL/PAYMENT.payment_id, payment_method, total, tax, delivery_option, bill_address, delivery_address, shipping_fee, order_status, delivery_date)

Step 4

BILL_PAYMENT(CUSTOMER.customer_id, SAVED_CARD.card_num, payment_id, payment_method, total, tax, delivery_option, bill_address, delivery_address, shipping_fee)

CUSTOMER_SERVICES (CUSTOMER.customer_id, service_id, assistant_no, contact_date, response_date, contact_type)

CATEGORY (category_id, category_name, parent_category_id)

Step 5

CART_ITEM (CUSTOMER.customer_id, SELLER_PRODUCT.product_id,SELLER_PRODUCT.seller_id, amount, price)

LIST_ITEM (list_id, SELLER_PRODUCT.product_id, SELLER_PRODUCT.seller_id)

FAVORITE (SELLER_PRODUCT.product_id, SELLER_PRODUCT.seller_id, CUSTOMER.customer_id)

ORDER_ITEM (SELLER_PRODUCT.product_id, SELLER_PRODUCT.seller_id, ORDER.order_no, amount, price)

BELONGS_TO (PRODUCT.product_id, CATEGORY.category_id)

FOLLOWS (SELLER.seller_id, CUSTOMER.customer_id)

ASK (SELLER.seller_id, CUSTOMER.customer_id, question, response, response_date)

CUSTOMER_OFFER (CUSTOMER.customer_id, SPECIAL_OFFER.offer_id)

DEAL_WITH (LOGISTIC.logistic_id, SELLER.seller_id)

SENT_BY (ORDER.order_no, LOGISTIC.logistic_id, tracking_no, shipment_date)

Step 6

CUSTOMER_ADDRESS (CUSTOMER.customer_id, address)

IMAGE_URL(PRODUCT.product_id, image_url)

Step 7

-

Step 8

PRODUCT_REVIEW (CUSTOMER.customer_id, PRODUCT.product_id, Rdate, Rtitle, comment, rating)

SELLER_REVIEW (CUSTOMER.customer_id, SELLER.seller_id, Rdate, Rtitle, comment, rating)

CUSTOMER_PREMIUM (customer_id, customer_name, e-mail, address, password, is_2factor_active, giftcard_balance, subscription_fee, renewal_date, start_date)

2nd ITERATION

Step 1 –

Step 2 –

Step 3

PAID_ORDERS (ORDER.order_no, BILL/PAYMENT.payment_id, CUSTOMER.customer_id, payment_method, total, tax, delivery_option, bill_address, delivery_address, shipping_fee, order_status, delivery_date)

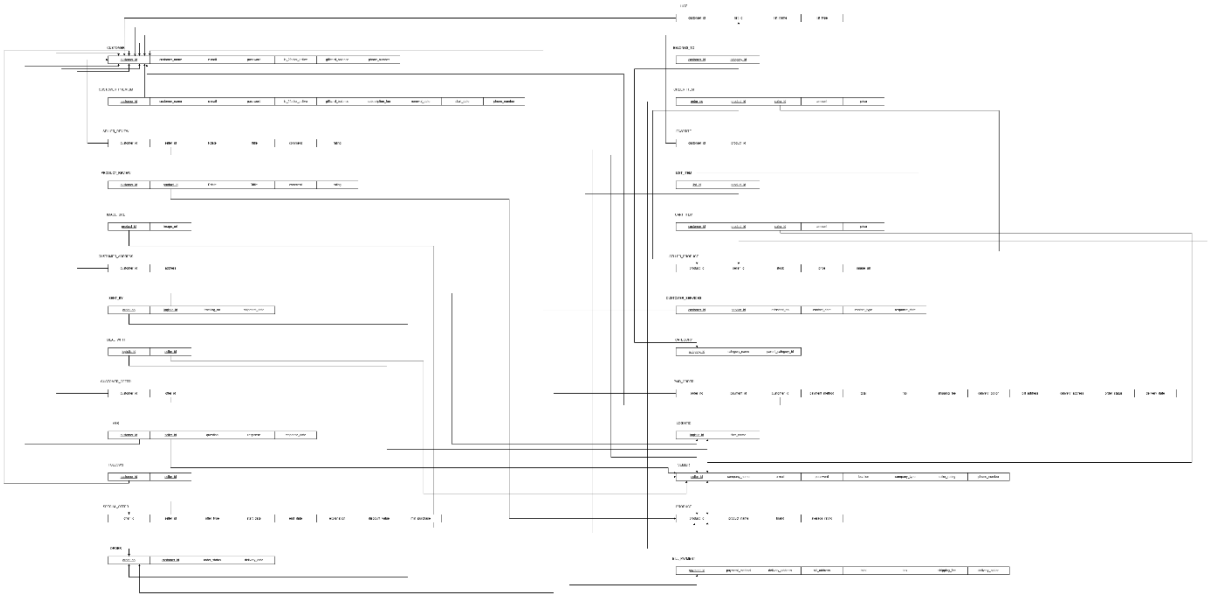
Step 4 –

Step 5 –

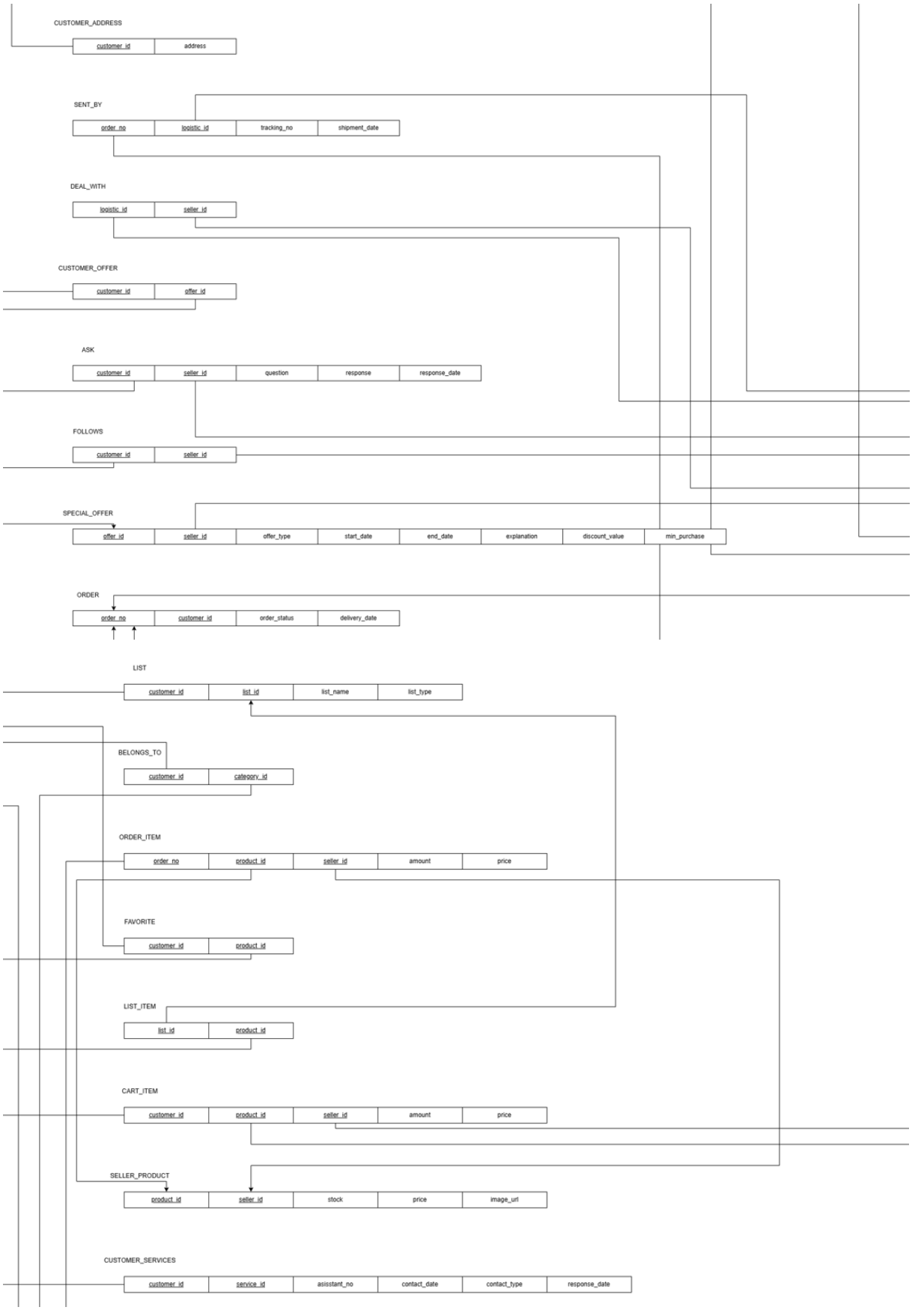
Step 6 –

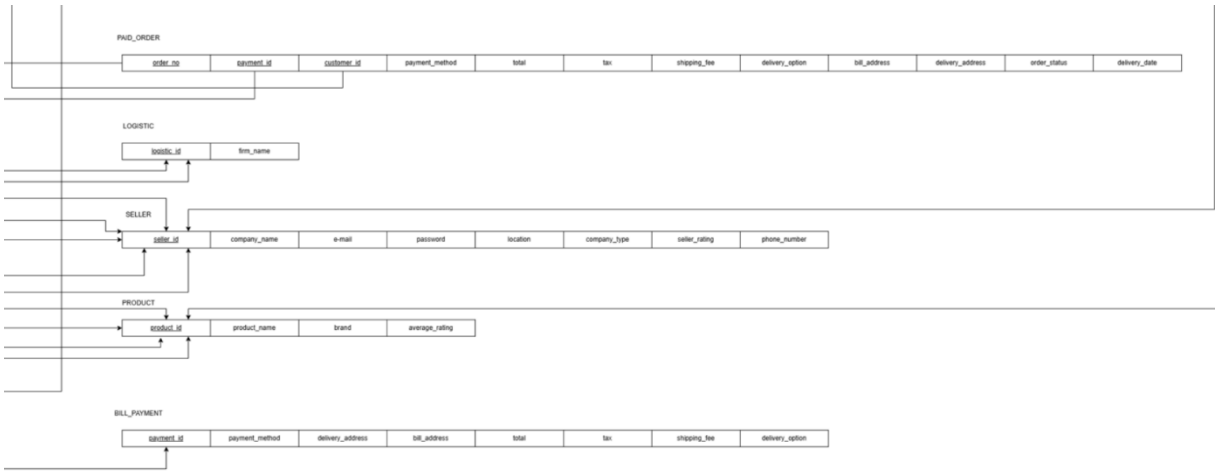
Step 7 –

Step 8 –




CUSTOMER
customer_id
customer_name
e-mail
password
is_2factor_active
giftcard_balance
phone_number
CUSTOMER_PREMIUM
customer_id
customer_name
e-mail
password
is_2factor_active
giftcard_balance
subscription_fee
renewal_date
start_date
phone_number
SELLER_REVIEW
customer_id
seller_id
Rdate
Rtitle
comment
rating
PRODUCT_REVIEW
customer_id
product_id
Rdate
Rtitle
comment
rating
IMAGE_URL
product_id
image_url
CUSTOMER_ADDRESS
customer_id
address


CUSTOMER_ADDRESS
customer_id
address
SENT_BY
order_no
logistic_id
tracking_no
shipment_date
DEAL_WITH
logistic_id
seller_id
CUSTOMER_OFFER
customer_id
offer_id
ASK
customer_id
seller_id
question
response
response_date
FOLLOWS
customer_id
seller_id
SPECIAL_OFFER
offer_id
seller_id
offer_type
start_date
end_date
explanation
discount_value
min_purchase
ORDER
order_no
customer_id
order_status
delivery_date
LIST
customer_id
list_id
list_name
list_type
BELONGS_TO
customer_id
category_id
ORDER_ITEM
order_no
product_id
seller_id
amount
price
FAVORITE
customer_id
product_id
LIST_ITEM
list_id
product_id
CART_ITEM
customer_id
product_id
seller_id
amount
price
SELLER_PRODUCT
product_id
seller_id
stock
price
image_url
CUSTOMER_SERVICES
customer_id
service_id
asisstant_no
contact_date
contact_type
response_date



IMPLEMENTATION-PYHSICAL MODEL

DDL STATEMENTS

```sql
-- 1. CATEGORY
CREATE TABLE CATEGORY (
    category_id INT NOT NULL,
    category_name VARCHAR(100) NOT NULL,
    parent_category_id INT,
    PRIMARY KEY (category_id),
    FOREIGN KEY (parent_category_id) REFERENCES CATEGORY(category_id)
);

-- 2. CUSTOMER
CREATE TABLE CUSTOMER (
    customer_id INT NOT NULL,
    customer_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    is_2factor_active BOOLEAN,
    giftcard_balance DECIMAL(10, 2),
    phone_number VARCHAR(15),
    PRIMARY KEY (customer_id)
);

-- 3. CUSTOMER_PREMIUM
CREATE TABLE CUSTOMER_PREMIUM (
    customer_id INT NOT NULL,
    customer_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    is_2factor_active BOOLEAN,
    giftcard_balance DECIMAL(10, 2),
    subscription_fee DECIMAL(10, 2),
    renewal_date DATE,
    start_date DATE,
    phone_number VARCHAR(15),
    PRIMARY KEY (customer_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);
```

```sql
-- 4. SELLER
CREATE TABLE SELLER (
    seller_id INT NOT NULL,
    company_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    password VARCHAR(255) NOT NULL,
    location VARCHAR(255),
    company_type VARCHAR(50),
    seller_rating DECIMAL(3, 2),
    phone_number VARCHAR(15),
    PRIMARY KEY (seller_id)
);

-- 5. PRODUCT
CREATE TABLE PRODUCT (
    product_id INT NOT NULL,
    product_name VARCHAR(100) NOT NULL,
    brand VARCHAR(50),
    average_rating DECIMAL(3, 2),
    PRIMARY KEY (product_id)
);

-- 6. IMAGE_URL
CREATE TABLE IMAGE_URL (
    product_id INT NOT NULL,
    image_url VARCHAR(255) NOT NULL,
    PRIMARY KEY (product_id, image_url),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);
```

```sql
-- 7. SELLER_PRODUCT
CREATE TABLE SELLER_PRODUCT (
    product_id INT NOT NULL,
    seller_id INT NOT NULL,
    stock INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (product_id, seller_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);

-- 8. CUSTOMER_ADDRESS
CREATE TABLE CUSTOMER_ADDRESS (
    customer_id INT NOT NULL,
    address TEXT NOT NULL,
    PRIMARY KEY (customer_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);

-- 9. CUSTOMER_SERVICES
CREATE TABLE CUSTOMER_SERVICES (
    customer_id INT NOT NULL,
    service_id INT NOT NULL,
    assistant_no INT NOT NULL,
    contact_date DATE NOT NULL,
    contact_type VARCHAR(50),
    response_date DATE,
    PRIMARY KEY (customer_id, service_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);
```

```sql
-- 10. SPECIAL_OFFER
CREATE TABLE SPECIAL_OFFER (
    offer_id INT NOT NULL,
    seller_id INT NOT NULL,
    offer_type VARCHAR(50),
    start_date DATE,
    end_date DATE,
    explanation TEXT,
    discount_value INT NOT NULL,
    min_purchase INT,
    PRIMARY KEY (offer_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);

-- 11. CUSTOMER_OFFER
CREATE TABLE CUSTOMER_OFFER (
    customer_id INT NOT NULL,
    offer_id INT NOT NULL,
    PRIMARY KEY (customer_id, offer_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (offer_id) REFERENCES SPECIAL_OFFER(offer_id)
);

-- 12. FAVORITE
CREATE TABLE FAVORITE (
    customer_id INT NOT NULL,
    product_id INT NOT NULL,
    PRIMARY KEY (customer_id, product_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);
```

```sql
-- 13. FOLLOWS
CREATE TABLE FOLLOWS (
    customer_id INT NOT NULL,
    seller_id INT NOT NULL,
    PRIMARY KEY (customer_id, seller_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);

-- 14. ASK
CREATE TABLE ASK (
    customer_id INT NOT NULL,
    seller_id INT NOT NULL,
    question TEXT NOT NULL,
    response TEXT,
    response_date DATE,
    PRIMARY KEY (customer_id, seller_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);

-- 19. LOGISTIC
CREATE TABLE LOGISTIC (
    logistic_id INT NOT NULL,
    firm_name VARCHAR(100) NOT NULL,
    PRIMARY KEY (logistic_id)
);

-- 15. DEAL_WITH
CREATE TABLE DEAL_WITH (
    logistic_id INT NOT NULL,
    seller_id INT NOT NULL,
    PRIMARY KEY (logistic_id, seller_id),
    FOREIGN KEY (logistic_id) REFERENCES LOGISTIC(logistic_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);
```

```sql
-- 16. SELLER_REVIEW
CREATE TABLE SELLER_REVIEW (
    customer_id INT NOT NULL,
    seller_id INT NOT NULL,
    Rdate DATE NOT NULL,
    Rtitle VARCHAR(100),
    comment TEXT,
    rating DECIMAL(3, 2),
    PRIMARY KEY (customer_id, seller_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);

-- 17. PRODUCT_REVIEW
CREATE TABLE PRODUCT_REVIEW (
    customer_id INT NOT NULL,
    product_id INT NOT NULL,
    Rdate DATE NOT NULL,
    Rtitle VARCHAR(100),
    comment TEXT,
    rating DECIMAL(3, 2),
    PRIMARY KEY (customer_id, product_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);

-- 18. CART_ITEM
CREATE TABLE CART_ITEM (
    customer_id INT NOT NULL,
    product_id INT NOT NULL,
    seller_id INT NOT NULL,
    amount INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (customer_id, product_id, seller_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);
```

```sql
-- 24. ORDER
CREATE TABLE `ORDER` (
    order_no INT NOT NULL,
    customer_id INT NOT NULL,
    order_status VARCHAR(50),
    delivery_date DATE,
    PRIMARY KEY (order_no),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);

-- 20. SENT_BY
CREATE TABLE SENT_BY (
    order_no INT NOT NULL,
    logistic_id INT NOT NULL,
    tracking_no VARCHAR(50) NOT NULL,
    shipment_date DATE NOT NULL,
    PRIMARY KEY (order_no, logistic_id),
    FOREIGN KEY (order_no) REFERENCES `ORDER`(order_no),
    FOREIGN KEY (logistic_id) REFERENCES LOGISTIC(logistic_id)
);

-- 21. LIST
CREATE TABLE LIST (
    customer_id INT NOT NULL,
    list_id INT NOT NULL,
    list_name VARCHAR(100) NOT NULL,
    list_type VARCHAR(50),
    PRIMARY KEY (list_id),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);
```

```sql
-- 22. LIST_ITEM
CREATE TABLE LIST_ITEM (
    list_id INT NOT NULL,
    product_id INT NOT NULL,
    PRIMARY KEY (list_id, product_id),
    FOREIGN KEY (list_id) REFERENCES LIST(list_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id)
);

-- 23. BELONGS_TO
CREATE TABLE BELONGS_TO (
    product_id INT NOT NULL,
    category_id INT NOT NULL,
    PRIMARY KEY (product_id, category_id),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id),
    FOREIGN KEY (category_id) REFERENCES CATEGORY(category_id)
);

-- 25. ORDER_ITEM
CREATE TABLE ORDER_ITEM (
    order_no INT NOT NULL,
    product_id INT NOT NULL,
    seller_id INT NOT NULL,
    amount INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (order_no, product_id, seller_id),
    FOREIGN KEY (order_no) REFERENCES `ORDER`(order_no),
    FOREIGN KEY (product_id) REFERENCES PRODUCT(product_id),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id)
);
```

```sql
-- 27. BILL/PAYMENT
CREATE TABLE `BILL/PAYMENT` (
    payment_id INT NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    total DECIMAL(10, 2) NOT NULL,
    tax DECIMAL(10, 2) NOT NULL,
    shipping_fee DECIMAL(10, 2),
    bill_address VARCHAR(255),
    delivery_address VARCHAR(255),
    delivery_option VARCHAR(50),
    PRIMARY KEY (payment_id)
);

-- 26. PAID_ORDER
CREATE TABLE PAID_ORDER (
    payment_id INT NOT NULL,
    order_no INT NOT NULL,
    customer_id INT NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    total DECIMAL(10, 2) NOT NULL,
    tax DECIMAL(10, 2) NOT NULL,
    shipping_fee DECIMAL(10, 2),
    delivery_option VARCHAR(50),
    bill_address VARCHAR(255),
    delivery_address VARCHAR(255),
    order_status VARCHAR(50),
    delivery_date DATE,
    PRIMARY KEY (payment_id, order_no, customer_id),
    FOREIGN KEY (payment_id) REFERENCES `BILL/PAYMENT`(payment_id),
    FOREIGN KEY (order_no) REFERENCES `ORDER`(order_no),
    FOREIGN KEY (customer_id) REFERENCES CUSTOMER(customer_id)
);
```

POPULATE THE DATABASE -INSERT

```sql
-- 1. CATEGORY
INSERT INTO CATEGORY (category_id, category_name, parent_category_id) VALUES
(1, 'Electronics', NULL),
(2, 'Home Appliances', 1),
(3, 'Mobile Phones', 1),
(4, 'Cosmetic', NULL),
(5, 'skin care', 4),
(6, 'make-up', 4);

-- 2. CUSTOMER
INSERT INTO CUSTOMER (customer_id, customer_name, email, password, is_2factor_active, giftcard_balance, phone_number) VALUES
(101, 'Murat Osman Ünal□r', 'muratosman@gmail.com', 'password123', TRUE, 0, '+905467891513'),
(102, 'Bahrihan Torpil', 'bahrihant9@gmail.com', 'bahri123', FALSE, 75.00, '+905530058989'),
(103, 'Kutlu Çağan Ak□n', 'cagankutlu@gmail.com', 'bahri123', FALSE, 75.00, '+905329201273'),
(104, 'Mehmet Ali Avc□', 'mali.avci121@gmail.com', 'avci01', TRUE, 300.00, '+905373265280');

-- 3. CUSTOMER_PREMIUM
INSERT INTO CUSTOMER_PREMIUM (customer_id, customer_name, email, password, is_2factor_active, giftcard_balance, subscription_fee, renewal_date, start_d
(101, 'Mehmet Ali Avc□', 'mali.avci121@gmail.com', 'avci01', TRUE, 300, 10.00, '2025-01-01', '2024-01-01', '05373265280');

-- 4. SELLER
INSERT INTO SELLER (seller_id, company_name, email, password, location, company_type, seller_rating, phone_number) VALUES
(201, 'Tech World', 'info@techworld.com', 'd2h18ydh', 'New York', 'Electronics', 0, '5551234567'),
(202, 'Home Goods', 'support@homegoods.com', '9d12k29fjd', 'Los Angeles', 'Home Appliances', 0, '5559876543'),
(203, 'Gratis', 'info@gratis.com', 'asdnfdsj123-12', 'Türkiye', 'Cosmetic', 0, '5551234567');

-- 5. PRODUCT
INSERT INTO PRODUCT (product_id, product_name, brand, average_rating) VALUES
(301, 'Smartphone', 'Apple', 0),
(302, 'Laptop', 'Lenovo', 0),
(303, 'Laptop', 'Dell', 0),
(304, 'Lipstick', 'Pastel', 0),
(305, 'Lipbalm', 'Nivea', 0),
(306, 'Washing Machine', 'Arçelik', 0);
```

```sql
-- 5. PRODUCT
INSERT INTO PRODUCT (product_id, product_name, brand, average_rating) VALUES
(301, 'Smartphone', 'Apple', 0),
(302, 'Laptop', 'Lenovo', 0),
(303, 'Laptop', 'Dell', 0),
(304, 'Lipstick', 'Pastel', 0),
(305, 'Lipbalm', 'Nivea', 0),
(306, 'Washing Machine', 'Arçelik', 0);

-- 6. IMAGE_URL
INSERT INTO IMAGE_URL (product_id, image_url) VALUES
(301, 'https://Apple.com/Apple1.jpg'),
(301, 'https://Apple.com/Apple2.jpg'),
(302, 'https://Lenovo.com/legion1.jpg'),
(303, 'https://Arçelik.com/ArçelikWM1.jpg'),
(304, 'https://Pastel.com/pastel1.jpg'),
(305, 'https://Nivea.com/nivea1.jpg'),
(302, 'https://Lenovo.com/legion3.jpg');

-- 7. SELLER_PRODUCT
INSERT INTO SELLER_PRODUCT (product_id, seller_id, stock, price) VALUES
(301, 201, 100, 500.00),
(301, 202, 100, 450.00),
(302, 201, 50, 1000.00),
(303, 202, 30, 300.00);

-- 8. CUSTOMER_ADDRESS
INSERT INTO CUSTOMER_ADDRESS (customer_id, address) VALUES
(101, '123 Main St, New York, NY'),
(102, '456 Elm St, Los Angeles, CA'),
(103, '789 Pine St, Chicago, IL');
```

```sql
-- 9. CUSTOMER_SERVICES
INSERT INTO CUSTOMER_SERVICES (customer_id, service_id, assistant_no, contact_date, contact_type, response_date) VALUES
(101, 1, 101, '2025-01-10', 'Email', '2025-01-11'),
(102, 1, 101, '2025-01-13', 'Message', '2025-01-15'),
(103, 2, 102, '2025-01-09', 'Phone', '2025-01-10');

-- 10. SPECIAL_OFFER
INSERT INTO SPECIAL_OFFER (offer_id, seller_id, offer_type, start_date, end_date, explanation, discount_value, min_purchase) VALUES
(401, 201, 'Discount', '2025-01-01', '2025-01-31', '10% off on all electronics', 40, NULL),
(402, 202, 'Coupon', '2025-01-15', '2025-02-15', 'Free shipping on orders above $50', 300, 2000);

-- 11. CUSTOMER_OFFER
INSERT INTO CUSTOMER_OFFER (customer_id, offer_id) VALUES
(101, 401),
(103, 401),
(102, 402);

-- 12. FAVORITE
INSERT INTO FAVORITE (customer_id, product_id) VALUES
(101, 301),
(102, 302);

-- 13. FOLLOWS
INSERT INTO FOLLOWS (customer_id, seller_id) VALUES
(101, 201),
(103, 201),
(102, 201),
(102, 202);

-- 14. ASK
INSERT INTO ASK (customer_id, seller_id, question, response, response_date) VALUES
(101, 201, 'Do you have more stock?', 'Yes, we restock next week.', '2025-01-12'),
(102, 202, 'Is there a warranty?', 'Yes, 1 year warranty is included.', '2025-01-13');
```

```sql
--  LOGISTIC
INSERT INTO LOGISTIC (logistic_id, firm_name) VALUES
(501, 'Aras Kargo'),
(502, 'Yurtiçi Kargo'),
(503, 'Trendyol Express');

-- 15. DEAL_WITH
INSERT INTO DEAL_WITH (logistic_id, seller_id) VALUES
(501, 201),
(502, 201),
(502, 202);

-- 16. SELLER_REVIEW
INSERT INTO SELLER_REVIEW (customer_id, seller_id, Rdate, Rtitle, comment, rating) VALUES
(101, 201, '2025-01-05', 'Great Service', 'The seller was very helpful.', 5.0),
(102, 201, '2025-01-08', 'absolute trash Service', 'The seller was very awful.', 2.0),
(102, 202, '2025-01-06', 'Quick Delivery', 'Fast shipping and well-packed.', 4.5);

-- 17. PRODUCT_REVIEW
INSERT INTO PRODUCT_REVIEW (customer_id, product_id, Rdate, Rtitle, comment, rating) VALUES
(101, 301, '2025-01-07', 'Excellent Phone', 'Great value for the price.', 4),
(102, 302, '2025-01-08', 'Amazing Laptop', 'Highly recommend it!', 3);

-- 18. CART_ITEM
INSERT INTO CART_ITEM (customer_id, product_id, seller_id, amount, price) VALUES
(101, 301, 201, 2, 1000.00),
(101, 302, 201, 2, 2000.00),
(101, 303, 201, 2, 3000.00),
(102, 302, 201, 1, 1000.00);
-- ORDER
INSERT INTO `ORDER` (order_no, customer_id, order_status, delivery_date) VALUES
(601, 101, 'Shipped', '2025-01-20'),
(602, 102, 'Processing', '2025-01-22'),
(603, 102, 'Returned', '2025-01-23');
```

```sql
-- 20. SENT_BY
INSERT INTO SENT_BY (order_no, logistic_id, tracking_no, shipment_date) VALUES
(601, 501, 'TRACK123', '2025-01-14'),
(602, 502, 'TRACK456', '2025-01-15');

-- 21. LIST
INSERT INTO LIST (customer_id, list_id, list_name, list_type) VALUES
(101, 701, 'Wish List', 'Public'),
(102, 702, 'Gift Ideas', 'Private');

-- 22. LIST_ITEM
INSERT INTO LIST_ITEM (list_id, product_id) VALUES
(701, 301),
(702, 302);

-- 23. BELONGS_TO
INSERT INTO BELONGS_TO (product_id, category_id) VALUES
(301, 1),
(306, 2),
(304, 4),
(303, 1),
(305, 4),
(306, 1),
(302, 1);
```

```sql
-- 25. ORDER_ITEM
INSERT INTO ORDER_ITEM (order_no, product_id, seller_id, amount, price) VALUES
(601, 301, 201, 3, 1500.00),
(602, 302, 201, 1, 1000.00);

-- 26. BILL/PAYMENT
INSERT INTO `BILL/PAYMENT` (payment_id, payment_method, total, tax, shipping_fee, bill_address, delivery_address, delivery_option) VALUES
(801, 'Credit Card', 1650.00, 100.00, 40.00, '123 Main St, New York, NY', '123 Main St, New York, NY', 'Standard'),
(802, 'Credit Card', 1150.00, 100.00, 50.00, '456 Elm St, Los Angeles, CA', '456 Elm St, Los Angeles, CA', 'Express'); -- express teslimat randevulu

-- 27. PAID_ORDER
INSERT INTO PAID_ORDER (payment_id, order_no, customer_id, payment_method, total, tax, shipping_fee, delivery_option, bill_address, delivery_address, orde
(801, 601, 101, 'Credit Card', 1650.00, 100.00, 50.00, 'Standard', '123 Main St, New York, NY', '123 Main St, New York, NY', 'Shipped', '2025-01-20'),
(802, 602, 102, 'PayPal', 1150.00, 100.00, 50.00, 'Express', '456 Elm St, Los Angeles, CA', '456 Elm St, Los Angeles, CA', 'Processing', '2025-01-22');
```

TRIGGERS

```sql
-- 1. TRIGGER: Update PRODUCT.average_rating when a new review is added to PRODUCT_REVIEW
DELIMITER $$
CREATE TRIGGER update_product_average_rating
AFTER INSERT ON PRODUCT_REVIEW
FOR EACH ROW
BEGIN
    DECLARE total_reviews INT;
    DECLARE total_rating DECIMAL(10, 2);

    -- Calculate the total number of reviews and the sum of ratings for the product
    SELECT COUNT(*) INTO total_reviews FROM PRODUCT_REVIEW WHERE product_id = NEW.product_id;
    SELECT SUM(rating) INTO total_rating FROM PRODUCT_REVIEW WHERE product_id = NEW.product_id;

    -- Update the average_rating in the PRODUCT table
    UPDATE PRODUCT
    SET average_rating = total_rating / total_reviews
    WHERE product_id = NEW.product_id;
END $$
DELIMITER ;

-- 2. TRIGGER: Update SELLER.seller_rating when a new review is added to SELLER_REVIEW
DELIMITER $$
CREATE TRIGGER update_seller_rating
AFTER INSERT ON SELLER_REVIEW
FOR EACH ROW
BEGIN
    DECLARE total_reviews INT;
    DECLARE total_rating DECIMAL(10, 2);

    -- Calculate the total number of reviews and the sum of ratings for the seller
    SELECT COUNT(*) INTO total_reviews FROM SELLER_REVIEW WHERE seller_id = NEW.seller_id;
    SELECT SUM(rating) INTO total_rating FROM SELLER_REVIEW WHERE seller_id = NEW.seller_id;

    -- Update the seller_rating in the SELLER table
    UPDATE SELLER
    SET seller_rating = total_rating / total_reviews
    WHERE seller_id = NEW.seller_id;
END $$
DELIMITER ;
```

```sql
-- 3. TRIGGER: Update stock in SELLER_PRODUCT when a product is added to CART_ITEM
DELIMITER $$
CREATE TRIGGER update_stock_on_cart_item_insert
AFTER INSERT ON CART_ITEM
FOR EACH ROW
BEGIN
    UPDATE SELLER_PRODUCT
    SET stock = stock - NEW.amount
    WHERE product_id = NEW.product_id AND seller_id = NEW.seller_id;
END $$
DELIMITER ;
```

CHECK CONSTRAINTS

```sql
ALTER TABLE SPECIAL_OFFER
ADD CONSTRAINT chk_min_purchase
CHECK (
    (offer_type = 'coupon' AND min_purchase IS NOT NULL AND min_purchase >= 0) OR
    (offer_type = 'discount')
);

ALTER TABLE SPECIAL_OFFER
ADD CONSTRAINT chk_discount_value
CHECK (
    discount_value > 0 AND
    (
        (offer_type = 'discount' AND discount_value < 100) OR
        (offer_type = 'coupon' AND discount_value <= min_purchase)
    )
);

ALTER TABLE SPECIAL_OFFER
ADD CONSTRAINT chk_dates
CHECK (
    start_date <= end_date
);

ALTER TABLE `ORDER`
ADD CONSTRAINT chk_order_status
CHECK (order_status IN ('Processing', 'Shipped', 'Delivered', 'Cancelled', 'Returned'));

ALTER TABLE SELLER_PRODUCT
ADD CONSTRAINT chk_price_stock
CHECK (price >= 0 AND stock >= 0);
```

INSERT-UPDATE-DELETE

```sql
-- INSERT Sorguları
INSERT INTO CART_ITEM (customer_id, product_id, seller_id, amount, price)
VALUES (102, 301, 201, 1, 450.00);

INSERT INTO SPECIAL_OFFER (offer_id, seller_id, offer_type, start_date, end_date, explanation, discount_value, min_purchase)
VALUES (403, 202, 'Coupon', '2025-02-01', '2025-02-28', '20% off on orders above $500', 20, 500);

INSERT INTO CUSTOMER_SERVICES (customer_id, service_id, assistant_no, contact_date, contact_type, response_date)
VALUES (104, 3, 103, '2025-01-16', 'Email', '2025-01-17');

-- DELETE Sorguları
DELETE FROM FAVORITE
WHERE customer_id = 101 AND product_id = 301;

DELETE FROM ORDER_ITEM
WHERE order_no = 602 AND product_id = 302 AND seller_id = 201;

DELETE FROM SPECIAL_OFFER
WHERE end_date < CURRENT_DATE;

-- UPDATE Sorguları
UPDATE CART_ITEM
SET amount = 3
WHERE customer_id = 101 AND product_id = 301 AND seller_id = 201;

UPDATE SELLER
SET seller_rating = 4.8
WHERE seller_id = 201;

UPDATE PRODUCT
SET average_rating = (SELECT AVG(rating) FROM PRODUCT_REVIEW WHERE product_id = 301)
WHERE product_id = 301;
```

SELECT STATEMENTS

```sql
-- Order history query sample
select * from `ORDER` as O
where O.order_status != 'pending' AND O.order_status != 'shipped';

-- Cart Total Info
SELECT
    customer_id,
    SUM(price * amount) AS total
FROM
    CART_ITEM
WHERE
    customer_id = 101
GROUP BY
    customer_id;


-- view seller ratings
SELECT
    S.seller_id,
    S.company_name,
    AVG(SR.rating) AS average_seller_rating
FROM
    SELLER S
LEFT JOIN
    SELLER_REVIEW SR ON S.seller_id = SR.seller_id
GROUP BY
    S.seller_id, S.company_name;
```

```sql
-- display the favorite list for the chosen customer
SELECT
    FAVORITE.customer_id,
    FAVORITE.product_id,
    PRODUCT.product_name,
    PRODUCT.brand
FROM
    FAVORITE
INNER JOIN
    PRODUCT
ON
    FAVORITE.product_id = PRODUCT.product_id
WHERE
    FAVORITE.customer_id = 101;


-- compares different sellers of the same product

SELECT
    SP.product_id,
    P.product_name,
    SP.seller_id,
    S.company_name,
    SP.price,
    SP.stock
FROM
    SELLER_PRODUCT SP
JOIN
    PRODUCT P ON SP.product_id = P.product_id
JOIN
    SELLER S ON SP.seller_id = S.seller_id
WHERE
    SP.product_id = 301;
```

```sql
-- view order item from chosen customer
SELECT
    O.order_no,
    O.order_status,
    O.delivery_date,
    P.product_name,
    OI.amount,
    OI.price
FROM
    `ORDER` O
JOIN
    ORDER_ITEM OI ON O.order_no = OI.order_no
JOIN
    PRODUCT P ON OI.product_id = P.product_id
WHERE
    O.customer_id = 101;


-- view chosen category product
SELECT
    C.category_name,
    P.product_name,
    P.brand,
    P.average_rating
FROM
    CATEGORY C
JOIN
    BELONGS_TO B ON C.category_id = B.category_id
JOIN
    PRODUCT P ON B.product_id = P.product_id
WHERE
    C.category_name = 'Cosmetic';
```

```sql
-- view sellers deal with their logistic
SELECT
    S.seller_id,
    S.company_name,
    L.logistic_id,
    L.firm_name
FROM
    SELLER S
JOIN
    DEAL_WITH DW ON S.seller_id = DW.seller_id
JOIN
    LOGISTIC L ON DW.logistic_id = L.logistic_id;

-- display special offers which are active
SELECT
    C.customer_id,
    C.customer_name,
    SO.offer_id,
    SO.offer_type,
    SO.explanation,
    SO.start_date,
    SO.end_date,
    SO.discount_value,
    SO.min_purchase
FROM
    CUSTOMER C
JOIN
    CUSTOMER_OFFER CO ON C.customer_id = CO.customer_id
JOIN
    SPECIAL_OFFER SO ON CO.offer_id = SO.offer_id
WHERE
    C.customer_id = 101 AND SO.end_date >= CURRENT_DATE;
```