Database Design

Etsy

Contents

Introduction

With the rise in online platforms for shopping, there are several products that cannot be found even on mainstream platforms like amazon and ebay. With creative minds coming together, there was a need for a marketplace to sell the innovations created by thousands of these creative minds. Personalization, art, craft, gifting and collectibles are some of the popular demands by consumers. Having to find a specific look for a particular gift and trying to make it personalized is not only a hard task but also archaic at the same time. Scrolling through a webpage/app is way easier when compared to having to go to craft stores for supplies and wandering around in different aisles for the required material.

With the increase in the online platforms and easy online payment methods adhered with secure methods, this task was made easier and more interesting by Etsy.

Etsy is an online shopping platform that has products ranging from personalized gifts for weddings, birthdays, anniversaries or any other occasion for that matter, to home decor including vintage looks. Important categories on the platform include jewelry, clothes, shoes, home & living, wedding & party, toys, entertainment, collectibles, art, craft supplies, gifts and gift cards. There are various sellers with different products, some produced in-house, because they are personalized, and some produced in higher quantities. Customers just search the type of product they are looking for and select the product that matches their needs from thousands of options. From here on the process works like any other online-shopping platform. There is a checkout cart, after which the user pays for their purchase and enters the remaining shipping details including the address for delivery, post which delivery handlers like FedEx and UPS take over and get it delivered to your doorstep without you having to worry about anything. There are two entities that make ends meet on the platform: Buyer and Seller.

Buyer/Customer :

A customer has the option to scroll through thousands of sellers and their products. As mentioned before, after the selection of products as per their needs, they pay and wait for the product to arrive. More importantly, pre-checkout they have the option to customize their products. For example, a mug can be customized to have a name on it, and a phone case can be customized to have a family portrait on the back of it.

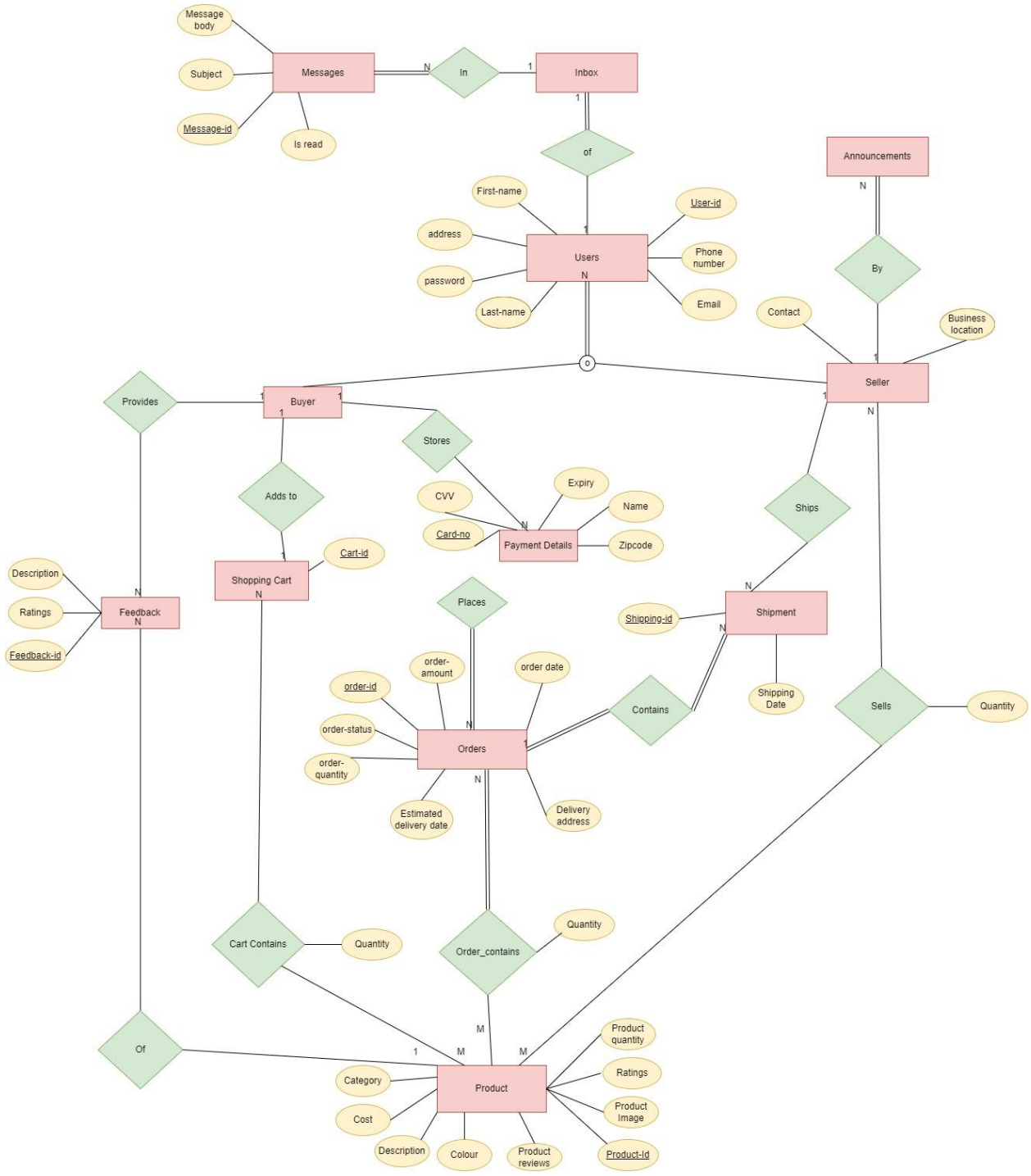
Seller:

The seller works on making their sub-store look good within the types of products they sell. The more customizations they offer, the more are the chances for getting products sold, and hence higher profits. They can make announcements about their upcoming and newly launched products, so the customers are aware of them. Sellers are in no way associated with Etsy. Most of the sellers are just using Etsy as a platform for an easy approach towards customers.

Requirements

For the user to have access to the platform from either seller or buyer ends, they have to have an account with Etsy. The user may require user-id, password, first and last name and an email address. Without an account they can scroll and search through the platform, however they would not be able to sell or buy any product. At any specific point in time, it is also necessary for the user to decide if they are currently acting as a buyer or a seller. The buyer also is required to have their payment details saved into the system for the ease of payments which includes the generic card details including the card number, cvv, expiration date and the name on the card. It is also important and necessary for the buyer to have entered a delivery address without which the service would be left incomplete. For the specific products within each seller, there can be a required customization setting for the buyer to enter. For example, if a coffee mug is customizable, it might need specifications for the customization. If they choose to not enter it, they can also move forward with the default version of the product. As per our system, any product that the buyer plans on purchasing, the product first has to make it to the cart. There is no way that the product can be simply bought without it first landing into the cart on the platform. At the arrival of the product the user is notified in their messages about the arrival of the delivery. A feedback is not necessarily required, but the user is notified about that too after some time of usage and delivery.

Enhanced Entity-Relation Diagram

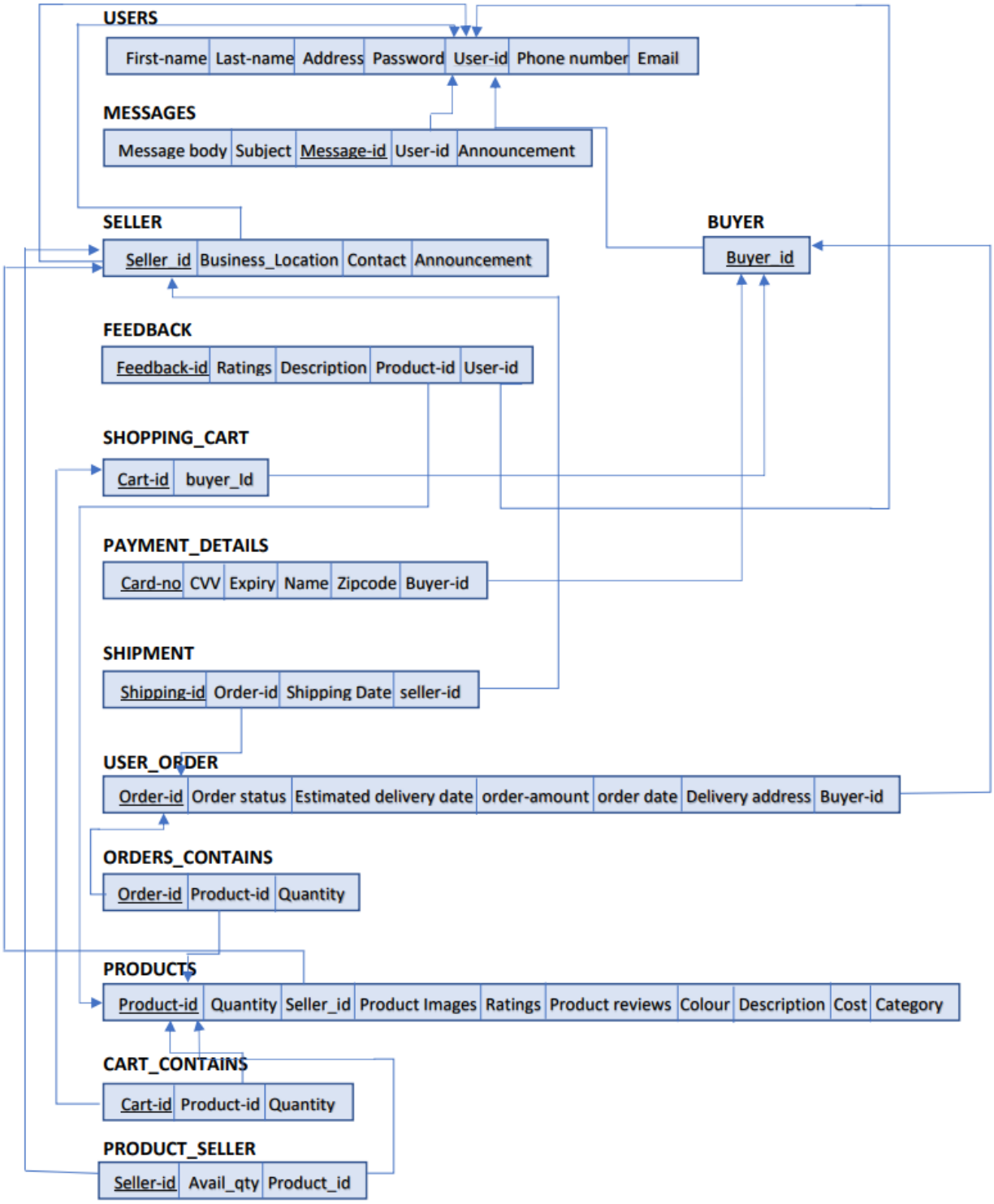


Basic Explanation of the EER

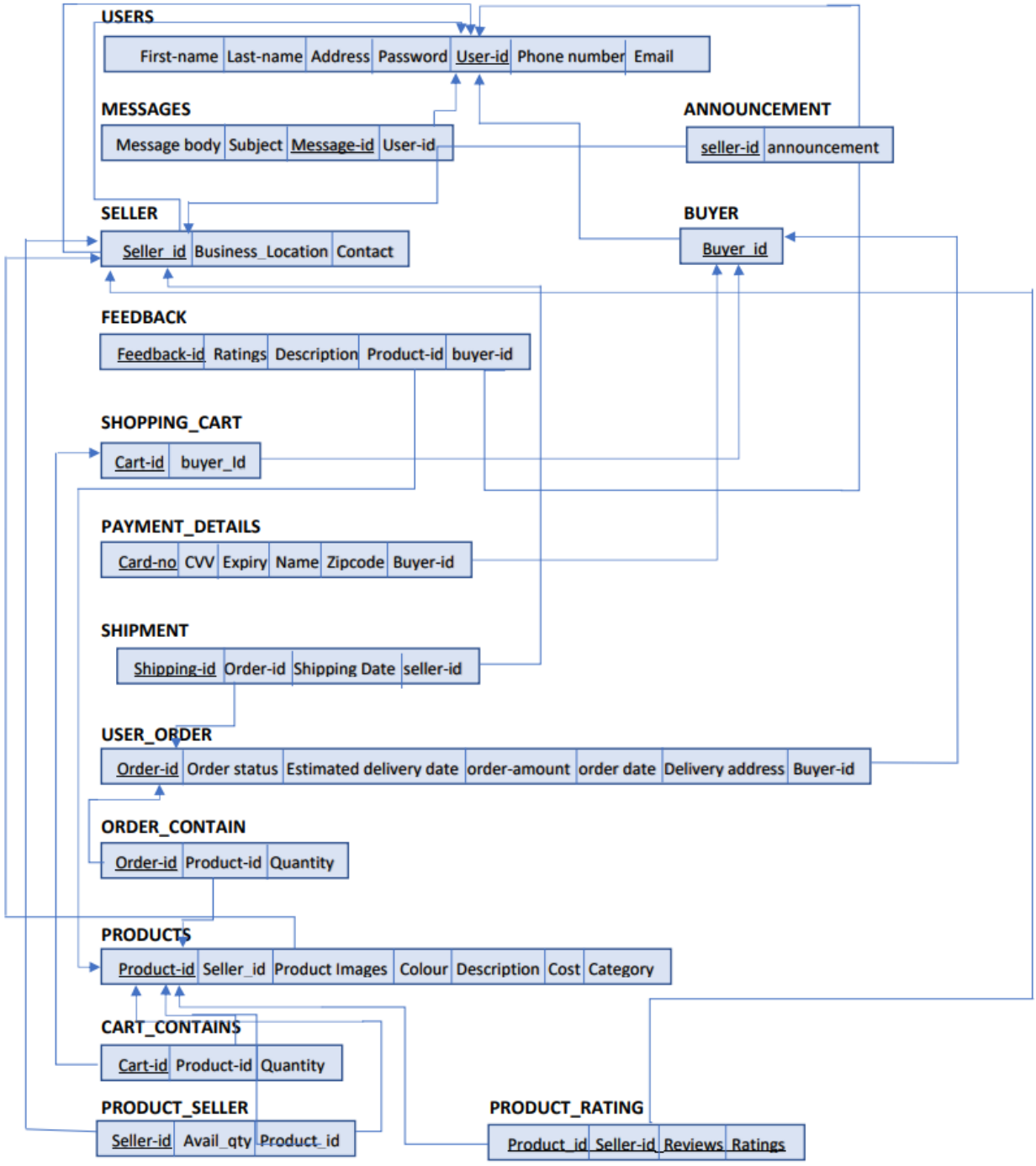
- A user can be categorized as either a buyer or a seller.
- A user has attributes like first name, last name, userid, email, phone and address.
- A user regardless of their category can have N number of messages in their inbox. Messages has attributes like subject, message body, message ID and user ID.
- A user can only have one inbox, and hence it's a 1:1 relationship.
- A seller has attributes like business location and contact, and furthermore, can sell N number of different products.
- A seller can make N number of announcements.
- A seller can ship N number of shipments requested by the buyer, and that order may contain N number of orders, which allows for multiple orders within the same shipment.
- Each order has attributes like order amount, status, date, quantity, estimated delivery date.
- An order can have N number of products within it.
- A buyer can add products one by one to the shopping cart, where the cart can contain N number of products.
- A buyer can also store N number of payment details, which has attributes like card number, cvv, expiration date, and the name on the card.
- A buyer can place N number of orders, where each order can contain N number of products as mentioned before.
- A buyer can provide feedback for multiple products that they have purchased, which has attributes like description of the feedback, ratings, product-id.

Mapping EER Diagram to Relational Schema

Pre-Normalization:



Post-Normalization:



Normalization

The following dependency exists in the Schema.

USERS(User-Id, First-name, Last-name, Address, Password, Phone number, Email)
User-Id → First-name, Last-name, Address, Password, Phone number, Email
USER table is already in 1NF,2NF and 3NF no need to normalize

MESSAGES(Message-id, User-id, Subject, Message body)
Message-id → User-id, Subject, Message body
MESSAGES table is already in 1NF,2NF and 3NF no need to normalize

SELLER(Seller-id, Business_Location, Contact, Announcement)
Seller_id → business_location, contact
Announcement(seller_id, announcement)
Seller_id → announcement
SELLER table has achieved 1NF,2NF and 3NF

FEEDBACK(Feedback-id, Ratings, Description, Product-id, User-id)
Feedback-id → Ratings, Description, Product-id, User-id
FEEDBACK table is already in 1NF,2NF and 3NF no need to normalize

SHOPPING CART(Cart-id, buyer_id)
Cart -id → buyer_id
SHOPPING CART table is already in 1NF,2NF and 3NF no need to normalize

PAYMENT DETAILS(Card-no, CVV, Expiry, Name, Zipcode, buyer-id)
Card-no → CVV, Expiry Name, Zipcode, buyer-id
PAYMENT DETAILS table is already in 1NF,2NF and 3NF no need to normalize

SHIPMENT(Shipping-id, Order-id, Shipping Date, seller-id)
Shipping-id → Order-id Shipping Date seller-id)
SHIPMENT table is already in 1NF,2NF and 3NF no need to normalize

ORDERS(Order-id, Order status, order-quantity, Estimated delivery date, order-amount, order date, Delivery address, buyer-id)
Order-id → Order status, order-quantity, Estimated delivery date, order-amount, order date, Delivery address, buyer-id
ORDERS table is already in 1NF,2NF and 3NF no need to normalize

ORDER_LIST(Order-id, Product-id, Quantity)
Order-id → Product-id, Quantity
ORDER_LIST table is already in 1NF,2NF and 3NF no need to normalize

PRODUCT(Product-id, Quantity, Product Images, Ratings, Product reviews, Colour, Description, Cost, Category)
Product-id → Quantity, Product Images, Ratings, Product reviews, Colour, Description, Cost, Category
PRODUCT table is not in 1NF
Reviews and Rating can have multiple values
To make it in 1NF create another table of ratings and reviews.
product_rating(product_id, seller_id, rating, reviews)

Now product table will be PRODUCT(Product-id, Quantity, Product Images, Colour, Description, Cost, Category)

CART_LIST(Cart-id, Product-id, Quantity)
Cart-id → Product-id, Quantity
CART_LIST table is already in 1NF,2NF and 3NF no need to normalize

SQL STATEMENTS

```
-- Queries to Create Databases based on normalized the ER diagram
CREATE TABLE users (
   user_Id VARCHAR(10) PRIMARY KEY,
   email VARCHAR(25),
   first_name VARCHAR(25) NOT NULL,
   last_name VARCHAR(25),
   phone_number NUMBER(10),
   user_address VARCHAR(50),
   User_password VARCHAR(30) NOT NULL
);

CREATE TABLE messages(
   message_id VARCHAR(10) PRIMARY KEY,
   message_body VARCHAR(255),
   subject VARCHAR(100),
   user_id VARCHAR(10) NOT NULL,
   FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

CREATE TABLE buyer(
   buyer_id VARCHAR(10) PRIMARY KEY,
   FOREIGN KEY (buyer_id) REFERENCES users(user_id) ON DELETE CASCADE
);

CREATE TABLE seller(
   seller_id VARCHAR(10) PRIMARY KEY,
   business_location VARCHAR(255),
   contact NUMBER(10),
   FOREIGN KEY (seller_id) REFERENCES users(user_id) ON DELETE CASCADE

);

CREATE TABLE announcement(
```

```
    announcement VARCHAR (255) NOT NULL,
    seller_id VARCHAR(10),
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id) ON DELETE CASCADE

);

CREATE TABLE payment_details(
    card_no INTEGER ,
    expiry DATE NOT NULL,
    cvv NUMBER(3) NOT NULL,
    zipcode NUMBER(5) NOT NULL,
    card_name VARCHAR(255) NOT NULL,
    buyer_id VARCHAR(10) NOT NULL,
    FOREIGN KEY (buyer_id) REFERENCES BUYER (buyer_id) ON DELETE CASCADE,
    PRIMARY KEY (card_no,buyer_id)
);

CREATE TABLE products(
    product_id VARCHAR(10) PRIMARY KEY,
    seller_id VARCHAR(10) NOT NULL,
    description VARCHAR(50),
    color VARCHAR(10),
    category VARCHAR(20),
    price NUMBER(10) NOT NULL,
    FOREIGN KEY (seller_id) REFERENCES SELLER(seller_id) ON DELETE CASCADE
);

CREATE TABLE user_order(
    order_id VARCHAR(10) PRIMARY KEY,
    order_status VARCHAR(20) NOT NULL,
    delivery_date DATE,
    order_date DATE DEFAULT CURRENT_DATE,
    -- Quantity INTEGER NOT NULL CHECK (Quantity >= 0),
    Delivery_address VARCHAR(255) NOT NULL,
    buyer_id VARCHAR(10) NOT NULL,
    FOREIGN KEY (buyer_id) REFERENCES BUYER (buyer_id) ON DELETE CASCADE
```

```
);

CREATE TABLE order_contain(
   order_id VARCHAR(10) NOT NULL,
   product_id VARCHAR(10) NOT NULL,
   order_quantity number DEFAULT 1,
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE,
   FOREIGN KEY (order_id) REFERENCES user_order(order_id) ON DELETE CASCADE,
   PRIMARY KEY (order_id,product_id)
);

CREATE TABLE shipment(
   shipping_id INTEGER PRIMARY KEY,
   order_id VARCHAR(10) NOT NULL,
   shipping_date DATE NOT NULL,
   seller_id VARCHAR(10) NOT NULL,
   FOREIGN KEY (seller_id) REFERENCES seller (seller_id) ON DELETE CASCADE,
   FOREIGN KEY (order_id) REFERENCES user_order(order_id) ON DELETE CASCADE
);


CREATE TABLE feedback(
   feedback_id VARCHAR(10) PRIMARY KEY,
   buyer_id VARCHAR(10) NOT NULL,
   description VARCHAR(50),
   product_id VARCHAR(10) NOT NULL,
   FOREIGN KEY (buyer_id) REFERENCES buyer (buyer_id) ON DELETE CASCADE,
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
);

CREATE TABLE shopping_cart(
   cart_id VARCHAR(10) PRIMARY KEY,
   buyer_id VARCHAR(10) NOT NULL,
   FOREIGN KEY (buyer_id) REFERENCES BUYER(buyer_id) ON DELETE CASCADE
);
```

```
CREATE TABLE cart_contains(
   cart_id VARCHAR(10) NOT NULL,
   product_id VARCHAR(10) NOT NULL,
   quantity NUMBER(10) DEFAULT 1,
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE,
   FOREIGN KEY (cart_id) REFERENCES SHOPPING_CART(cart_id) ON DELETE CASCADE
);

CREATE TABLE product_seller(
   product_id VARCHAR(10) NOT NULL,
   seller_id VARCHAR(10) NOT NULL,
   avail_qty VARCHAR(255) DEFAULT 1,
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE,
   FOREIGN KEY (seller_id) REFERENCES seller(seller_id) ON DELETE CASCADE
);

CREATE TABLE product_rating(
   product_id VARCHAR(10) NOT NULL,
   seller_id VARCHAR(10) NOT NULL,
   reviews VARCHAR(255),
   rating VARCHAR(255) CHECK (RATING>= 0 AND RATING<=5),
   FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE,
   FOREIGN KEY (seller_id) REFERENCES seller(seller_id) ON DELETE CASCADE
);



INSERT INTO USERS VALUES('1','niki@gmail', 'Miki', 'Joseph',1234567890, 'Texas', 'abc123');
INSERT INTO USERS VALUES('2','abc@gmail', 'Ross', 'John',1234567892, 'Texas', 'abc1233');

INSERT INTO MESSAGES VALUES('1A', 'about Product', 'Product', '1');
INSERT INTO MESSAGES VALUES('1B', 'About Product', 'Product', '2');

INSERT INTO SELLER VALUES('1', 'dallas',9876543210);
INSERT INTO SELLER VALUES('2', 'Austin',9870543210);
```

```
INSERT INTO ANNOUNCEMENT VALUES('ANNOUNCEMENT', '1');
INSERT INTO ANNOUNCEMENT VALUES('ANNOUNCEMENT2', '2');

INSERT INTO BUYER VALUES('1');
INSERT INTO BUYER VALUES('2');

INSERT INTO payment_details VALUES(12345, date '2024-02-09',123, 12345, '123456789', '1');

INSERT INTO PRODUCTS VALUES('67', '1', 'PRODUCT DESCRIPTION', 'RED', '5', 23);
INSERT INTO PRODUCTS VALUES('68', '2', 'PRODUCT ', 'BLUE', '9', 230);

INSERT INTO user_order VALUES('ASD1', 'COMPLETED', DATE '2022-03-03', DATE
'2022-03-05','Richardson','2');
INSERT INTO user_order VALUES('PQR2', 'COMPLETED', DATE '2022-03-02', DATE
'2022-03-08','Richardson','1');

INSERT INTO SHIPMENT VALUES(9,'ASD', DATE '2022-03-02','1');
INSERT INTO SHIPMENT VALUES(8,'PQR', DATE '2022-03-02','2');

INSERT INTO ORDER_CONTAIN VALUES('ASD','68');
INSERT INTO ORDER_CONTAIN VALUES('PQR','67');

INSERT INTO FEEDBACK VALUES('MNB','1', 'DESCRIPTION','67');
INSERT INTO FEEDBACK VALUES('MNV','2', 'DESCRIPTIONS','68');

INSERT INTO SHOPPING_CART VALUES('456','1');
INSERT INTO SHOPPING_CART VALUES('459','2');
INSERT INTO SHOPPING_CART VALUES('013','13');

INSERT INTO cart_contains VALUES('456','67',2);
INSERT INTO cart_contains VALUES('459','68',4);
INSERT INTO cart_contains VALUES('013','67',3);

INSERT INTO PRODUCT_SELLER VALUES('67','1',10);
INSERT INTO PRODUCT_SELLER VALUES('68','2',20);
```

Procedures

1. **Registering a buyer**

This procedure helps the buyer register themselves as a user on the platform. For that the user needs to input information such as email, userID, first name, last name, phone number and any other personal details requested by the system.

```
CREATE
OR REPLACE PROCEDURE registerBuyer (
   user_Id IN VARCHAR,
   email IN VARCHAR,
   first_name IN VARCHAR,
   last_name IN VARCHAR,
   phone_number IN NUMBER,
   user_address in VARCHAR,
   pswd IN VARCHAR
) AS BEGIN
INSERT INTO
   USERS
VALUES
   (
      user_Id,
      email,
      first_name,
      last_name,
      phone_number,
      user_address,
      pswd

   );
INSERT INTO
   BUYER
VALUES
   (
      user_Id
   );
```

```
END registerBuyer;
```

2. Registering a seller

This procedure helps the seller to register themselves as a user on the platform. For that the seller needs to input information such as email, userID, first name, last name, phone number and any other personal details requested by the system. This is similar to registering a buyer, however there is some extra set of information and setup needed for the seller.

```
CREATE
OR REPLACE PROCEDURE registerSeller(
   user_Id IN VARCHAR,
   email IN VARCHAR,
   first_name IN VARCHAR,
   last_name IN VARCHAR,
   phone_number IN NUMBER,
   user_address in VARCHAR,
   pswd IN VARCHAR,
   business_location IN VARCHAR,
   user_contact IN NUMBER
) AS BEGIN
INSERT INTO
   USERS
VALUES
   (
      user_Id,
      email,
      first_name,
      last_name,
      phone_number,
      user_address,
      pswd
   );
INSERT INTO
```

```
    SELLER
VALUES
    (
        user_Id,
        business_location,
        user_contact
    );

END registerSeller;
```

```
BEGIN
 REGISTERSELLER(
'12','miki1@gmail','Miki1','Joseph1',1334567890,'Tbexas','abc123','TEXAS',1234567894);
END;
```

```
BEGIN
 registerBuyer( '13','niki13@gmail','Miki3','Joseph3',1334567890,'TCexas','abc123');
END;
```

3. Placing order from the cart

This procedure helps in establishing the information required to keep the process going in terms of the cart. It takes the items present in the cart, and processes them for accountability of purchase within the database. Attributes like order date, delivery address and any other details about the entire order from the cart is taken care of.

```
CREATE
OR REPLACE PROCEDURE place_order_from_cart (
    v_orderId IN VARCHAR,
    v_buyerId IN VARCHAR,
    v_deliveryDate IN DATE,
    v_delivery_address IN VARCHAR
) AS
```

```
v_listPrice NUMBER := 0;
v_cartTotal NUMBER := 0;

CURSOR prodId IS
SELECT
   cart_contains.product_id, CART_CONTAINS.QUANTITY
FROM
   CART_CONTAINS
   INNER JOIN shopping_cart ON cart_contains.cart_id = shopping_cart.cart_id
WHERE
   buyer_id = v_buyerId;

v_pId VARCHAR(255);
v_cart_quantity number;

BEGIN
   INSERT INTO
   USER_ORDER
VALUES
   (
      v_orderId,
      'Order Placed',
      v_deliveryDate,
      sysdate,
      v_delivery_address,
      v_buyerId
   );
   OPEN prodId;

LOOP
   FETCH prodId INTO v_pId,v_cart_quantity;

EXIT
WHEN (prodId % notfound);
```

```
SELECT
   price INTO v_listPrice
FROM
   PRODUCTS
WHERE
   product_id = v_pId;

v_cartTotal := (v_cartTotal + (v_listPrice * v_cart_quantity));

INSERT INTO
   ORDER_CONTAIN
VALUES
   (v_orderId, v_pId,v_cart_quantity);

END LOOP;

CLOSE prodId;

dbms_output.put_line(v_cartTotal);


END place_order_from_cart;


BEGIN
   place_order_from_cart('123456', 13, date '2024-02-09', 'TEXAS');
END;
```

Trigger

1. **Updating Quantity on the sellers end**

This trigger basically takes care of the updation of the quantity on the seller side. It updates it to the new quantity available with the seller after a buyer has made a purchase for that product. For example, if a seller has 10 mugs to sell and a buyer purchases 2 of them, the new quantity that the seller now has should be 8, which is what this trigger takes care of.

```
CREATE
OR REPLACE TRIGGER updateQuantity
AFTER
INSERT
   ON order_contain
   FOR EACH ROW
DECLARE
v_productId INTEGER;
v_quantity INTEGER;

CURSOR prodId IS
SELECT
   PRODUCT_ID, ORDER_QUANTITY
FROM
   ORDER_CONTAIN
WHERE
   order_id = :new.order_id;

BEGIN OPEN prodId;

LOOP FETCH prodId INTO v_productId,v_quantity;

EXIT
WHEN (prodId %notfound);

UPDATE
   product_seller
SET
   AVAILABE_QUANTITY = AVAILABE_QUANTITY - v_quantity
WHERE
   product_id = v_productId;
```

```
END LOOP;

CLOSE prodId;

END updateQuantity;
```

2. Emptying the cart post order

This trigger makes sure that the cart is empty after the order has been placed by the buyer. It wouldn't make sense if the items were still in the cart even after the order was placed, which is what this trigger takes care of.

```
CREATE
OR REPLACE TRIGGER emptyCart
AFTER
INSERT
  ON user_order
  FOR EACH ROW
  DECLARE
  cartID varchar(255);
BEGIN
  select cart_id into cartID
  from shopping_cart
  where shopping_cart.BUYER_ID = :new.BUYER_ID;

DELETE FROM
  shopping_cart
WHERE
  shopping_cart.BUYER_ID = :new.BUYER_ID;

DELETE FROM
  CART_CONTAINS
WHERE
  cart_id = cartID;

END;
```

Conclusion

Etsy is an american e-commerce marketplace, mainly centered towards homemade, vintage, and craft supplies. It serves as a convenience for several occasions and needs around the country. With that high of a responsibility, there is a need for special attention by every seller for every product because of the presence of customizations.

In this particular Database project, our team reverse engineered the database system of Etsy with special attention to detail for all of its functionalities to the most closely possible outcomes compared to the real database system. We understood and analyzed various aspects of the design involved in the creation of the e-commerce marketplace that we today know as Etsy. Furthermore, we enhanced our understanding of the creation of the database system in real time, including its normalization, leading to having a lucid and conceptual image about real time databases and the design that comes with it. To conclude, the project was a great learning experience for the entire team because it helped us in having a greater comprehension of databases, their designs, and other concepts involved while working with a detailed database, inadvertently allowing us to understand the precautionary measure needed while working on designing a database.