Ebay Project Report

Team Members :

1) Tathya R. Thaker
   Department of Computer Science
   University of Texas at Dallas
   txt200018@utdallas.edu

2) Nishant J. Ramani
   Department of Computer Science
   University of Texas at Dallas
   nxr200011@utdallas.edu

3) Bhargaw R. Patel
   Department of Computer Science
   University of Texas at Dallas
   brp190004@utdallas.edu

Team Number - 3
Course Number - CS 6360
Section Number - 001
Project - Ebay 3

Table of Content

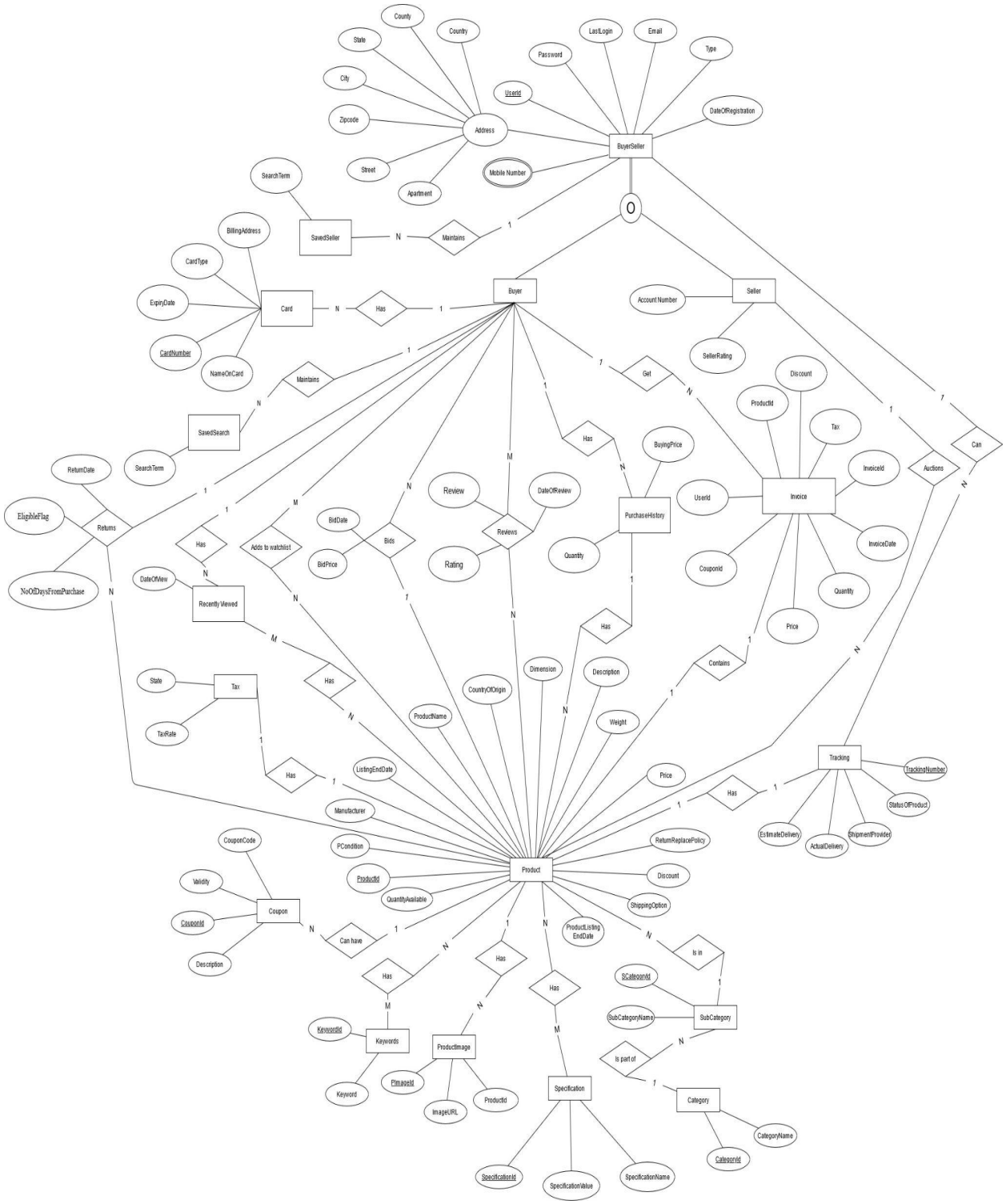
What Is eBay?

eBay is an American multinational e-commerce corporation based in San Jose, California, that facilitates consumer-to-consumer and business-to-consumer sales through its website. eBay was founded by Pierre Omidyar in 1995. eBay is a multibillion-dollar business with operations in about 32 countries, as of 2019. The company manages the eBay website, an online auction and shopping website in which people and businesses buy and sell a wide variety of goods and services worldwide. The website is free to use for buyers, but sellers are charged fees for listing items after a limited number of free listings, and again when those items are sold.

System Design:

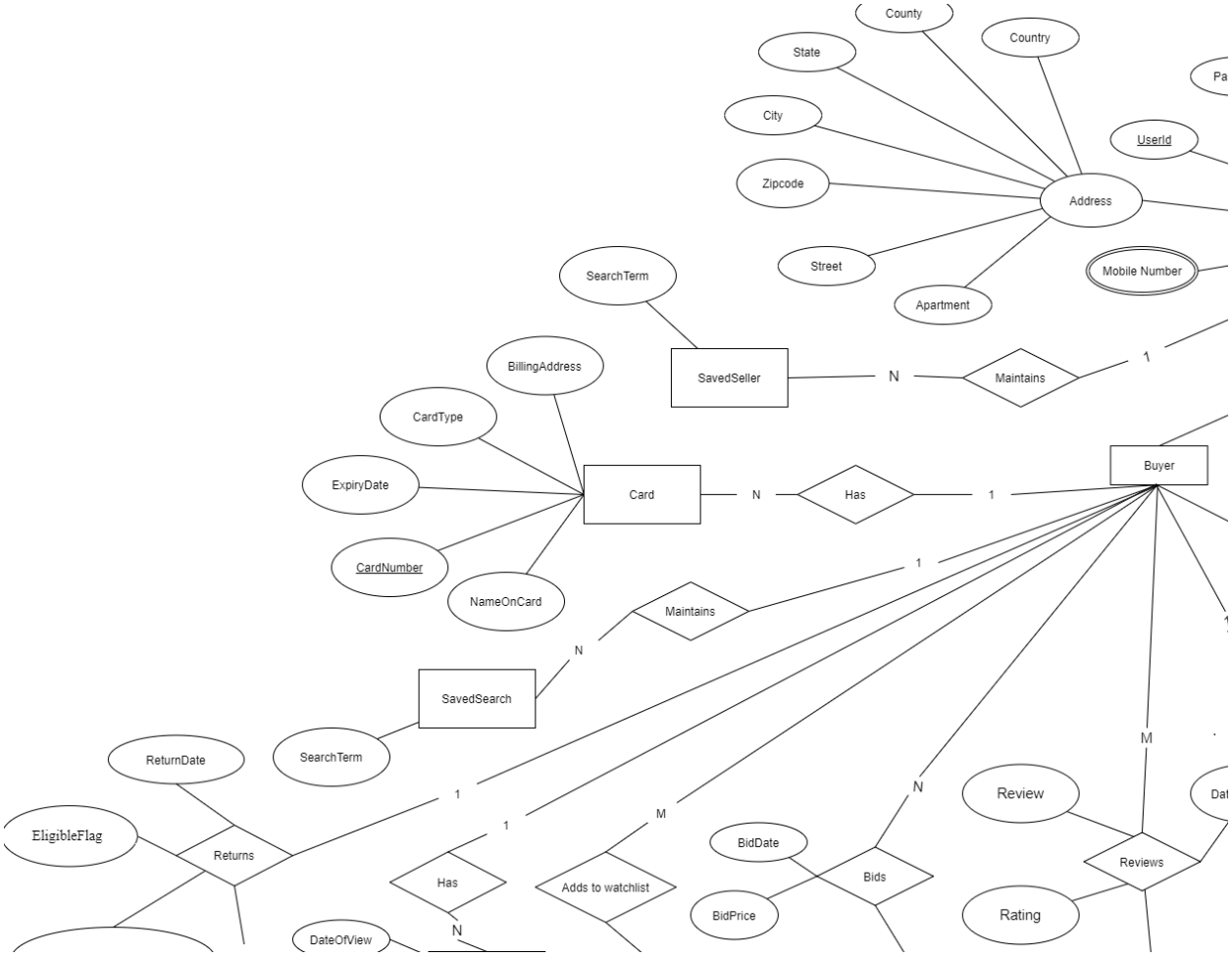
eBay is an online marketplace where users can buy and sell goods.
Buyers may purchase fresh or used goods that are marketed by specific customers. Few items are also available at auctions. The website lists the total number of bids on the product as well as the highest bid at the moment. The user will bid according to the prices shown, and the winner is determined after the timer runs out.

EBay EER Diagram

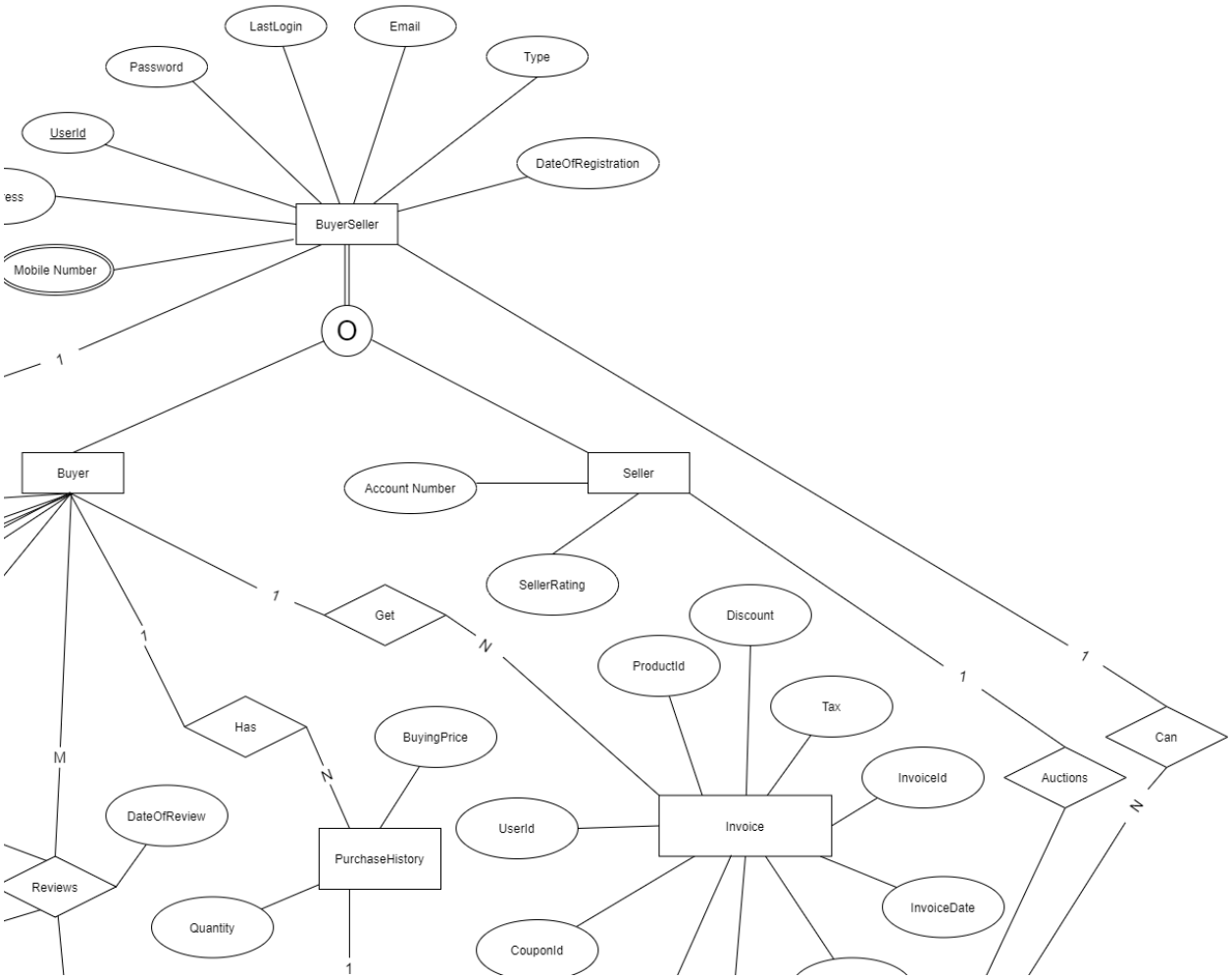


Link to EERD: https://drive.google.com/file/d/1Nbjs60LtN11Ba0fciepqdn_HUV6xt9uz/view?usp=sharing

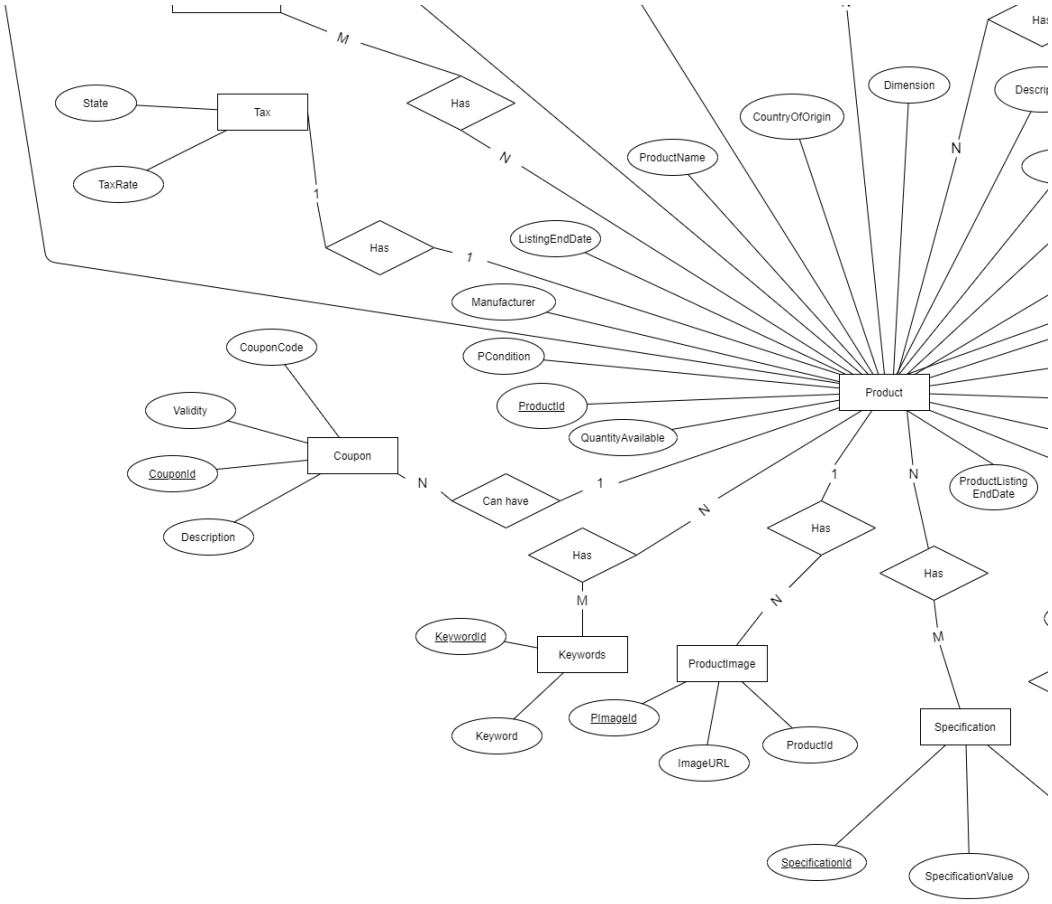
Top Left EER Diagram



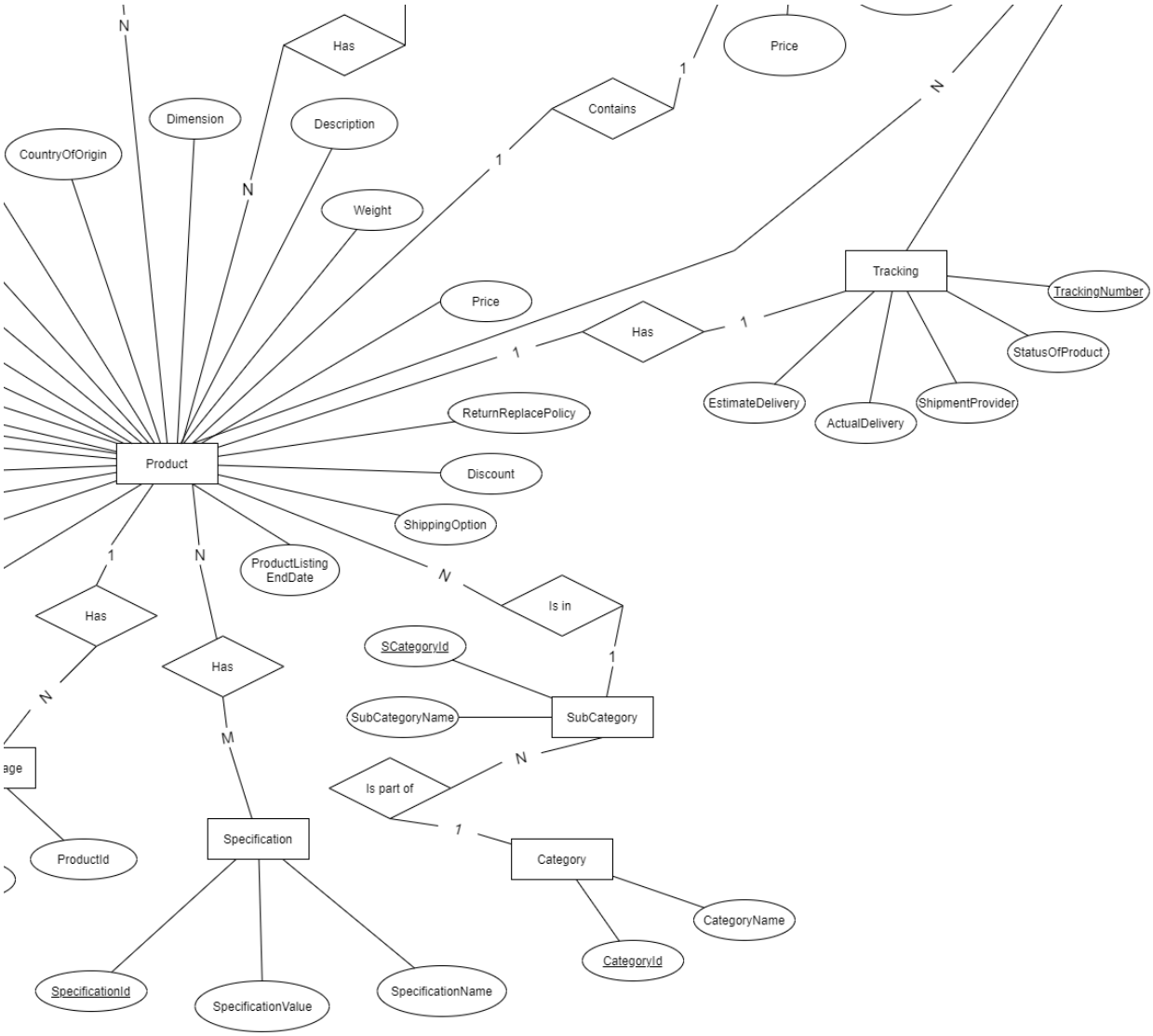
Top Right EER Diagram



Bottom Left EER Diagram



Bottom Right EER Diagram



Functional Requirement:

1. **A user can register** – A user can register as a buyer or a seller
2. **A buyer can bid on products** - A user can bid and the highest bidder purchases the product.
3. **A seller can add Listing end date** - A seller can add a date when the listing of product ends.
4. **A seller can add products** - A seller can sell product with details about product.
5. **A buyer can save a seller** - A buyer can save their favourite seller to get updates when they release a new product.
6. **A buyer can give review** - A buyer can write reviews and give rating to a product.
7. **A buyer can add products to Watch List** - A buyer can store the products which they plant to keep an eye on.
8. **A buyer can add card details** - A buyer can save card details to have a smooth buying experience.
9. **A buyer can add products to a shopping** - buyer add to the shopping cart the products they want to buy.
10. **Product can have keywords** - to help user find product better keywords
11. **Product has images** - to help user understand how product will look
12. **User have purchase history** - A user can see what they have bought in past
13. **A buyer can see recently viewed products** - A buyer can see what product they have opened in recent time.
14. **A seller can allow a coupon on product** - A seller can have a coupon code to offer discount on certain products
15. **A buyer can redeem a coupon** - A buyer can redeem a coupon on a selected product to get a discount.
16. **A user can have multiple addresses** - A user can order products to different locations without adding an address every time.
17. **A buyer can track the order** - A buyer can use tracking id to know the status of the product.
18. **A product has specification** - A seller can choose subcategory and for that category list of possible specifications can be written for the product.

Relationships:

1. **BuyerSeller - Product** = Each user can buy multiple products and each product can be sold and bought by multiple buyers. Cardinality is Many to Many.
2. **Address - BuyerSeller** = Each buyer has multiple Address, while each Address is associated to 1 buyer. Cardinality is 1 to many.
3. **Zip - Address** = Each address has 1 zip code, also each zip is associated with 1 Address. Cardinality is 1 to 1.
4. **Product - Coupon** = Each product can have many coupons, while each coupon is associated with 1 product. Cardinality is 1 to Many.
5. **Product - Specification** = Each Product can have multiple specifications. Also, each specification is associated with multiple products. Cardinality is many to many.
6. **Product - SubCategory** = Each product has one SubCategory. And each SubCategory is associated with one Product. Cardinality is Many to 1.
7. **Specification - SubCategory** = Each specification is associated to one SubCategory while each SubCategory can have multiple Specification. Cardinality is Many to 1.
8. **Product - Invoice** = Each Product has one invoice and every invoice is associated with one product. Cardinality is 1 to 1.
9. **SubCategory - Category** = Each SubCategory is associated with 1 Category and each category can have multiple sub categories. Cardinality is Many to one.
10. **Product - Keyword** = Each product can have multiple keywords and every keyword is associated with one product. Cardinality is Many to one.
11. **BuyerSeller - Watchlist** = Each buyer can add multiple products to watchlist. While, every product in watchlist is associated with multiple users. Cardinality is Many to Many.
12. **Product - PurchaseHistory** = Each Product will have one purchase history and one purchase history detail will be associated with many products. Cardinality is 1 to many.
13. **Product - Tracking** = Each product will have its own tracking details and every tracking record is associated with one product. Cardinality is 1 to 1.
14. **Product - Watchlist** = Each product can be added once in watchlist and watchlist can have multiple products. Cardinality is Many to 1.
15. **Tax- Invoice** = Each invoice has its tax and each tax is associated to one invoice. Cardinality is 1 to 1.
16. **Product - ProductRating** = Each product review is for one product and each product can have multiple reviews. Cardinality is Many to 1.

17. **BuyerSeller - PurchaseHistory** = Each buyer can have multiple products in its purchase history while every purchase history is associated with one buyer only. Cardinality is 1 to many.
18. **BuyerSeller - RecentlyViewed** = Each buyer can have multiple products in its recently viewed records while every product in that record is associated with one buyer only. Cardinality is 1 to Many.
19. **Tracking - Address** = Each tracking of product has one address assigned to it. While, each shipping address is associated with one tracking record. Cardinality is 1 to 1.
20. **Tracking - BuyerSeller** = Every buyer can have one tracking record for its products and each tracking record is associated with one buyer. Cardinality is 1 to 1.
21. **PhoneNo - BuyerSeller** = Every buyer seller has many phone numbers but each phone number is associated with one person. Cardinality is many to 1.
22. **SavedSeller - BuyerSeller** = Each seller can be saved by many users and each user can save multiple sellers. Cardinality is Many to Many.
23. **SavedSearch - BuyerSeller** = Each user can save multiple search terms. Each search term is associated with the user. Cardinality is 1 to Many.
24. **Card - BuyerSeller** = Each card is used by one user while every user can pay using different cards. Cardinality is 1 to Many.

1 to 1 Relationships = 6
1 to Many Relationships = 7
Many to 1 Relationships = 7
Many to Many Relationships = 4
Total Number of Relationships = 24

Normalizing the Existing Schema :

1-NF : A relation is in first normal form if and only if the domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain.

2-NF : A relation is in the second normal form if it fulfills the following two requirements:

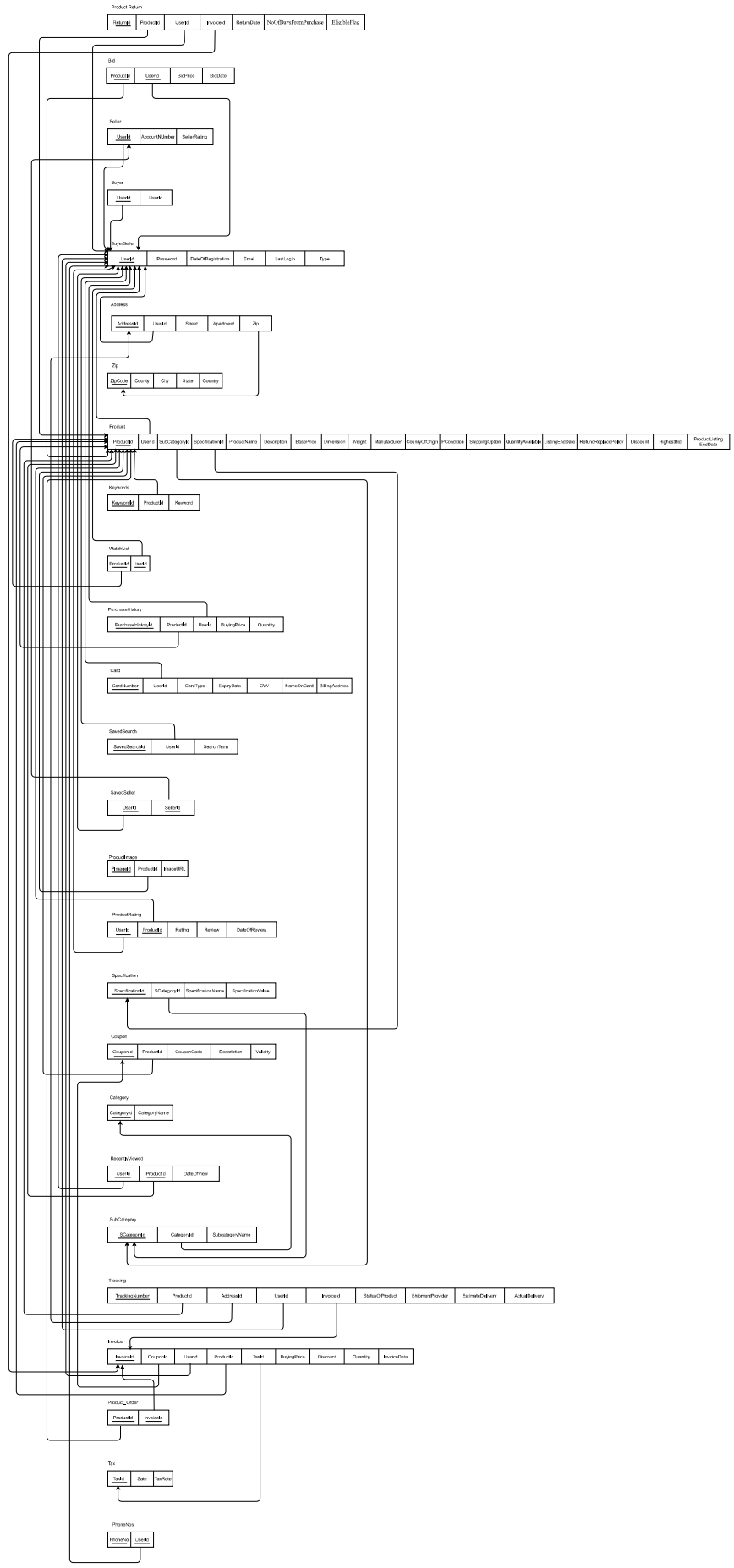
1. It is in first normal form.
2. It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

3 - NF : A database relation is said to meet third normal form standards if all the attributes are functionally dependent on solely the primary key. Codd defined this as a relation in second normal form where all non-prime attributes depend only on the candidate keys and do not have a transitive dependency on another key.

Our Relations are already Normalized

Link to Relationship Diagram:
https://drive.google.com/file/d/1efZxGu3zsbjt2iX-zRhz3xY7Ctnk8cKg/view?usp=sharing

Relational Tables:



Tables:

1) Buyer
2) Seller
3) Address
4) Zip
5) PhoneNos
6) Keywords
7) ProductImage
8) WatchList
9) PurchaseHistory
10) Card
11) SavedSearch
12) SavedSeller
13) ProductRating
14) RecentlyViewed
15) Category
16) Subcategory
17) Coupon
18) Specification
19) Tracking
20) Invoice
21) ReturnP
22) Bid

Primary Keys:

1) In the **BuyerSeller** table, we have **UserId** as Primary key.
2) In the **Address** table, we have **AddressId** as Primary key.
3) In the **Zip** table, we have **ZipCode** as the Primary key.
4) In the **PhoneNos** table, we have **PhoneNo** as Primary key.
5) In the **PurchaseHistory** table, we have **PurchaseHistoryId** as Primary key.
6) In the **Product** table, we have **productId** as Primary key.
7) In the **Keywords** table, we have **KeywordId** as Primary key.
8) In the **ProductImage** table, we have **PImageId** as Primary key.
9) In the **PurchaseHistory** table, we have **PurchaseHistoryId** as Primary key.
10) In the **Card** table, we have **CardNumber** as Primary key.
11) In the **SavedSearch** table, we have **SavedSearchId** as Primary key.
12) In the **Category** table, we have **CategoryId** as Primary key.
13) In the **SubCategory** table, we have **SCategoryId** as Primary key.
14) In the **Coupon** table, we have **CouponId** as Primary key.
15) In the **Specification** table, we have **SpecificationId** as Primary key.
16) In the **Tracking** table, we have **TrackingNumber** as Primary key.
17) In the **Invoice** table, we have **InvoiceId** as Primary key.
18) In the **Tax** table, we have **TaxId** as Primary key.

Foreign Key :

1) In the **Address** Table, we have **UserId** as foreign key.
2) In the **PhoneNos** Table, we have **UserId** as foreign key.
3) In the **Product** Table, we have **UserId** as foreign key.
4) In the **Product** Table, we have **SubCategoryId** as foreign key.
5) In the **Product** Table, we have **SpecificationId** as foreign key.
6) In the **Keywords** Table, we have **ProductId** as foreign key.
7) In the **ProductImage** Table, we have **ProductId** as foreign key.
8) In the **Watchlist** Table, we have **UserId** as foreign key.
9) In the **Watchlist** Table, we have **ProductId** as foreign key.
10) In the **PurchaseHistory** Table, we have **UserId** as foreign key.
11) In the **PurchaseHistory** Table, we have **ProductId** as foreign key.
12) In the **Card** Table, we have **UserId** as foreign key.
13) In the **SavedSearch** Table, we have **UserId** as foreign key.
14) In the **SavedSeller** Table, we have **UserId** as foreign key.
15) In the **ProductRating** Table, we have **UserId** as foreign key.

16) In the **ProductRating** Table, we have **ProductId** as foreign key.
17) In the **RecentlyViewed** Table, we have **UserId** as foreign key.
18) In the **RecentlyViewed** Table, we have **ProductId** as foreign key.
19) In the **SubCategory** Table, we have **CategoryId** as foreign key.
20) In the **Coupon** Table, we have **ProductId** as foreign key.
21) In the **Specification** Table, we have **SCategoryId** as foreign key.
22) In the **Tracking** Table, we have **UserId** as foreign key.
23) In the **Tracking** Table, we have **ProductId** as foreign key.
24) In the **Tracking** Table, we have **AddressId** as foreign key.
25) In the **Invoice** Table, we have **UserId** as foreign key.
26) In the **Invoice** Table, we have **CouponId** as foreign key.
27) In the **Invoice** Table, we have **ProductId** as foreign key.
28) In the **Invoice** Table, we have **TaxId** as foreign key.
29) In the **Bid** Table, we have **UserId** as foreign key.

SQL Query for Ebay Database:

CREATE TABLE Buyer (UserId int NOT NULL PRIMARY KEY, Password varchar(16), DateOfRegistration date, Email varchar(30), LastLogin date);

CREATE TABLE Seller (SellerId int NOT NULL PRIMARY KEY, Password varchar(16), DateOfRegistration date, Email varchar(30), AccountNumber int, LastLogin date, SellerRating int);

CREATE TABLE Address (AddressId int NOT NULL PRIMARY KEY, UserId int, Street varchar(30), Apartment varchar(30), Zip int);

CREATE TABLE Zip (ZipCode int NOT NULL PRIMARY KEY, County varchar(30), City varchar(30), State varchar(30), Country varchar(30));

CREATE TABLE PhoneNos (PhoneNo int NOT NULL PRIMARY KEY, UserId int);

CREATE TABLE Product (ProductId int NOT NULL PRIMARY KEY, UserId int, SubCategoryId int, SpecificationId int, ProductName varchar(30), Description varchar(255), BasePrice int, HighestBid int, Dimension varchar(30), Weight int, Manufacturer varchar(30), CountryOfOrigin varchar(30), PCondition int, ShippingOption int, QuantityAvailable int, LisitingEndDate Date, RefundReplacePolicy int, Discount int);

CREATE TABLE Keywords (KeywordId int NOT NULL PRIMARY KEY, ProductId int, Keyword varchar(30));

CREATE TABLE ProductImage (PImageId int NOT NULL PRIMARY KEY, ProductId int, ImageURL varchar(50));

CREATE TABLE WatchList (ProductId int, UserId int);

CREATE TABLE PurchaseHistory (PurchaseHistoryId int NOT NULL PRIMARY KEY, ProductId int, UserId int, BuyingPrice int, Quantity int);

CREATE TABLE Card (CardNumber int NOT NULL PRIMARY KEY, UserId int, CardType varchar(30), isDefault int, ExpiryDate date, CVV int, NameOnCard varchar(30), BillingAddress varchar(255));

```
CREATE TABLE SavedSearch (SavedSearchId int NOT NULL PRIMARY KEY,
    UserId int, SearchTerm varchar(30));

CREATE TABLE SavedSeller (UserId int, SellerId int);

CREATE TABLE ProductRating (UserId int, ProductId int, Rating int, Review
    varchar(255), DateOfReview date);

CREATE TABLE RecentlyViewed (UserId int, ProductId int, DateOfView  date);

CREATE TABLE Category (CategoryId int  NOT NULL PRIMARY KEY,
    CategoryName varchar(30));

CREATE TABLE SubCategory (SCategoryId int  NOT NULL PRIMARY KEY,
    CategoryId int, SubCategoryName varchar(30));

CREATE TABLE Coupon (CouponId int NOT NULL PRIMARY KEY, ProductId int,
    CouponCode varchar(10), Description varchar(255), Validity date);

CREATE TABLE Specification (SpecificationId int NOT NULL PRIMARY KEY,
    SCategoryId int, SpecificationName varchar(255), SpecificationValue
    varchar(255));

CREATE TABLE Tracking (TrackingNumber int NOT NULL PRIMARY KEY,
    ProductId int, AddressId int, UserId int, StatusOfProduct int, ShipmentProvider
    varchar(30), EstimateDelivery date, ActualDelivery date);

CREATE TABLE Invoice (InvoiceId int NOT NULL PRIMARY KEY, CouponId int,
    UserId int, ProductId int, TaxId int, Price int, Discount int, Quantity int,
    InvoiceDate date);

CREATE TABLE Tax (TaxId int NOT NULL PRIMARY KEY, TaxRate int, State
    varchar(30));

CREATE TABLE Bid (UserId int, ProductId int, BidPrice int, BidDate Date);

CREATE TABLE ReturnP (ReturnId int NOT NULL PRIMARY KEY, UserId int,
    ProductId int, InvoiceId int, ReturnDate Date, NoOfDaysFromPurchase int,
    EligibleFlag int );
```

```
ALTER TABLE `Address` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `PhoneNos` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Product` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Product` ADD FOREIGN KEY (`SubCategoryId` ) REFERENCES
    `SubCategory` (`SCategoryId`) ON DELETE CASCADE;

ALTER TABLE `Product` ADD  FOREIGN KEY (`SpecificationId`) REFERENCES
    `Specification`(`SpecificationId`) ON DELETE CASCADE;

ALTER TABLE `Keywords` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `ProductImage` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `WatchList` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `WatchList` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `PurchaseHistory` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `PurchaseHistory` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Card` ADD FOREIGN KEY (`UserId`) REFERENCES `BuyerSeller`
    (`UserId`) ON DELETE CASCADE;

ALTER TABLE `SavedSearch` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;
```

```
ALTER TABLE `SavedSeller` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `ProductRating` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `ProductRating` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `RecentlyViewed` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `RecentlyViewed` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Subcategory` ADD FOREIGN KEY (`CategoryId`) REFERENCES
    `Category` (`CategoryId`) ON DELETE CASCADE;

ALTER TABLE `Coupon` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Specification` ADD FOREIGN KEY (`SCategoryId`) REFERENCES
    `SubCategory` (`SCategoryId`) ON DELETE CASCADE;

ALTER TABLE `Tracking` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Tracking` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Tracking` ADD FOREIGN KEY (`AddressId`) REFERENCES
    `Address` (`AddressId`) ON DELETE CASCADE;

ALTER TABLE `Invoice` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Invoice` ADD FOREIGN KEY (`CouponId`) REFERENCES
    `Coupon` (`CouponId`) ON DELETE CASCADE;
```

```
ALTER TABLE `Invoice` ADD FOREIGN KEY (`ProductId`) REFERENCES
    `Product` (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Invoice` ADD FOREIGN KEY (`TaxId`) REFERENCES `Tax`
    (`TaxId`) ON DELETE CASCADE;

ALTER TABLE `Bid` ADD FOREIGN KEY (`UserId`) REFERENCES `BuyerSeller`
    (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Bid` ADD FOREIGN KEY (`ProductId`) REFERENCES `Product`
    (`ProductId`) ON DELETE CASCADE;

ALTER TABLE `Return` ADD FOREIGN KEY (`UserId`) REFERENCES
    `BuyerSeller` (`UserId`) ON DELETE CASCADE;

ALTER TABLE `Return` ADD FOREIGN KEY (`ProductId`) REFERENCES `Product`
    (`ProductId`) ON DELETE CASCADE;
```

Procedures:

1) To check whether the product is eligible for return or not.

```
CREATE OR REPLACE PROCEDURE CHECKRETURNELIGIBILE AS
CURSOR  C1
is
SELECT * FROM RETURNS;
BEGIN
            FOR R
            IN C1
            LOOP
                  IF X.NoOfDaysFromPurchase  > 60 THEN
                                    update RETURNS T
                              set EligibleFlag = 0
                              WHERE R.ReturnId=T.ReturnId;
                  END IF;
            END LOOP;
END;
```

2) To make product quantity 0 once the listing end date is reached as it will be no more for sale.

```
CREATE OR REPLACE PROCEDURE ProductStatus AS
CURSOR  C1
is
SELECT * FROM Product;
BEGIN
      FOR P
      IN C1
      LOOP
      IF SYSDATE  > P.LisitingEndDate THEN
                        update Product P1
                  set Quantity = 0
                        where P1.ProductId= P.ProductId;
      END IF;
      END LOOP;
END;
```

3) To set default card for the buyer

```
CREATE OR REPLACE PROCEDURE setDefaultCard (
   cardnumber    IN   INTEGER,
   userid_up   IN   VARCHAR)
AS BEGIN UPDATE Card SET isdefault = 1 WHERE UserId = userid_up AND
     CardNumber = cardnumber;
END setDefaultCard;
```

Triggers:

1) To update the seller rating

```
CREATE OR REPLACE TRIGGER updateSellerRatings AFTER
INSERT OR UPDATE OF SellerRating ON Seller
FOR EACH ROW
DECLARE
new_rating NUMBER(2, 1);
seller_id_to_update VARCHAR(255);
Count int = (SELECT COUNT(*) FROM PRODUCT P, SELLER S WHERE
    S.SELLERID = P.SELLERID)
BEGIN
new_rating := :new.rating;
SELECT
seller_id
INTO seller_id_to_update
FROM product WHERE product_id = :new.product_id;
UPDATE seller SET SellerRating = ( ( SellerRating * Count ) + new_rating ) / ( Count +
    1 ),
rating_count = rating_count + 1 WHERE seller_id = seller_id_to_update;
END;
```

2) To remove the seller who have less than 2 rating from the system

```
CREATE OR REPLACE TRIGGER remove_low_rated_seller
AFTER UPDATE ON SELLER FOR EACH ROW BEGIN
SELECT SellerRating FROM Seller;
if(SellerRating  < 2) then    delete from Seller where SellerRating <2;
END;
```

Project Implementation and its Results:

1) Buyer table registering buyer :

UserId	Password	DateOfRegistration	Email	LastLogin
1	nishant	2020-12-16	nishant@utd.com	2021-02-18
2	bhargaw	2021-01-16	bhargaw@utd.com	2021-04-14

2) Seller table registering seller :

SellerId	Password	DateOfRegistration	Email	AccountNumber	LastLogin	SellerRating
1	tathya	2020-05-26	tathya@utd.com	123456789	2021-03-09	5

3) Invoice table with invoice details:

InvoiceId	CouponId	UserId	ProductId	TaxId	Price	Discount	Quantity	InvoiceDate
20212	*NULL*	2	1	2	700	*NULL*	2	2021-04-20
121321	200	1	2	1	600	20	2	2021-05-03

4) Address table, having address details of buyer and seller:

AddressId	UserId	Street	Apartment	Zip
1	1	7575 Frankford	2114	78232

5) Coupon table, showing details of offers on products :

CouponId	ProductId	CouponCode	Description	Validity
200	1	20OFF	20 Discount	2021-05-08

6) ProductRating table which stores the rating given to products:

UserId	ProductId	Rating	Review	DateOfReview
1	2	5	Amazing Product	2021-05-03

7) Keyword table, assigning keywords to products :

KeywordId	ProductId	Keyword
1	1	Mobile
2	1	Accessories
3	1	Case
4	2	Car
5	2	Wind Shield

8) PurchaseHistoryId table shows the record of purchased items by seller:

PurchaseHistoryId	ProductId	UserId	BuyingPrice	Quantity
1	1	2	700	1
2	2	1	600	2

9) Tracking table helps to track the order :

TrackingNumber	ProductId	AddressId	UserId	StatusOfProduct	ShipmentProvider	EstimateDelivery	ActualDelivery
113235	1	1	2	0	USPS	2021-05-02	2021-05-05
46531763	2	1	1	1	FedEx	2021-05-03	2021-05-03