**PROJECT REPORT**

**Finished by : Eng / Mahmoud Mohamed Ahmed Hussain**

**E-mail : mahmoud.mohamed002@outlook.com**

In the project I have created an amazon e-commerce database system which fulfils basic functional requirements for an e-commerce website. The database system fulfils the following functional requirements -

1. **A user can register**– A user can be a buyer or a seller
2. **A user can place order**- each order contains multiple products.
3. **A seller can add products**- this contains the details of a product.
4. **A buyer can give review**- a user can write reviews about a product.
5. **A user can add products to a Wishlist**- user can store the products which they like but not yet ready to buy
6. **A user can add products to a shopping cart**- users add to the shopping cart the products they want to buy.
7. **Product can be of multiple category**- it defines the category of a product like clothing, electronics etc
8. **Product can be carried by different carrier**- the shipping service through which a product can be shipped.
9. **A user can have multiple address and contact details**- users address and phone number saved in the account
10. **A user can have multiple Card info**- users saved card in the account
11. **A buyer can add images to its review**- each review can have images associated with it
12. **A seller can add images to its product** - each product can have multiple images associated with it, so taken in separate table.

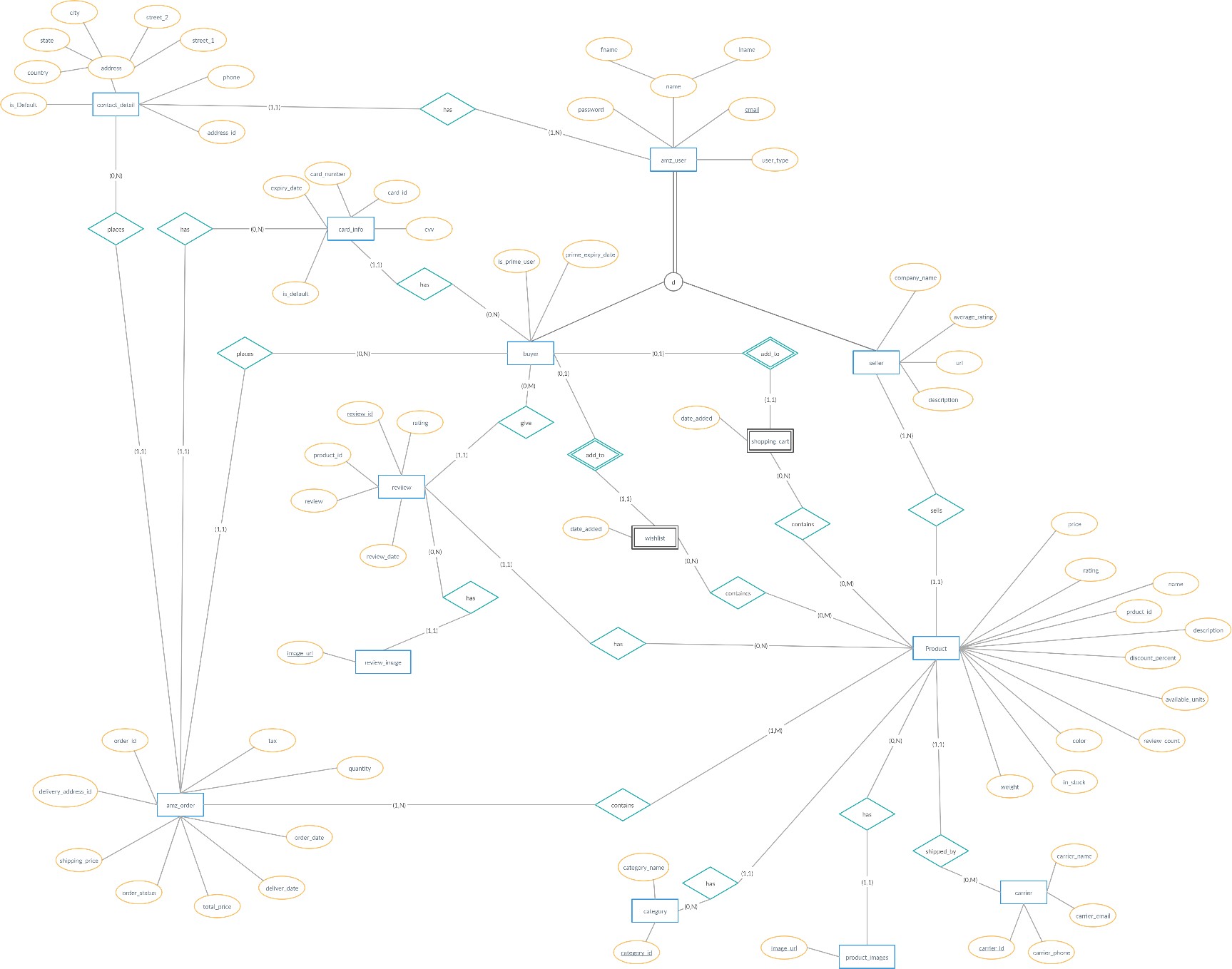
# RELATIONSHIPS

1. **User-contact\_details:** each user can have multiple contact details saved, while each contact detail will have only 1 user linked. Thus, cardinality is 1: N
2. **Buyer-card\_info:** each buyer can have many cards, while each card is associated with 1 buyer. Thus, cardinality is 1: N
3. **Buyer-order:** buyer places order. A buyer can place many orders, while each order is linked with only 1 buyer. Thus, cardinality is 1: N
4. **Order-product:** each order can contain many products, and each product can come in many orders. Thus, cardinality is M: N
5. **Seller-product:** seller sells products. Each seller can sell many products, while each product has only 1 seller. Thus, cardinality is 1: N
6. **Buyer-reviews:** buyer can write a review. Each buyer can write multiple reviews, while each review has only 1 buyer. Thus, cardinality is 1: N
7. **Review-products:** each review is for 1 product while each product can have many reviews. Thus, cardinality is 1: N
8. **Buyer-wishlist:** each buyer can have only 1 Wishlist, and each Wishlist has 1 buyer. Thus, cardinality is 1:1.
9. **Buyer- shopping cart:** each buyer has only 1 shopping cart while each shopping cart is associated with 1 buyer. Thus, cardinality is 1:1.
10. **Wishlist-product:** each Wishlist contains many products, and each product can be in many wish lists. Thus, cardinality is N:M
11. **Shopping\_cart-product:** each cart contains many products, and each product can be in many carts. Thus, cardinality is N:M
12. **Product-category:** each product has only 1 category while each category has many products. Thus, cardinality is 1: N
13. **Product-carrier:** each product is shipped by 1 carrier, while each carrier ships many products.
14. **Review-review\_images:** each review has many review images while each review image is linked with 1 review. Thus, cardinality is 1: N
15. **Product-product\_images:** each product has many product images while each product image is linked with 1 product. Thus, cardinality is 1: N
16. **Order-card\_info:** each order has a card linked with it, while each card can be used for many orders. Thus, cardinality is 1: N
17. **Order-contact details:** each order has 1 contact detail (address, phone), while each contact detail is linked with multiple orders. Thus, cardinality is 1: N
    * Number 1:1 relationship = 2
    * Number of N: M relationships = 3
    * Number of 1: N relationships = 12
    * Total Relationships = 17

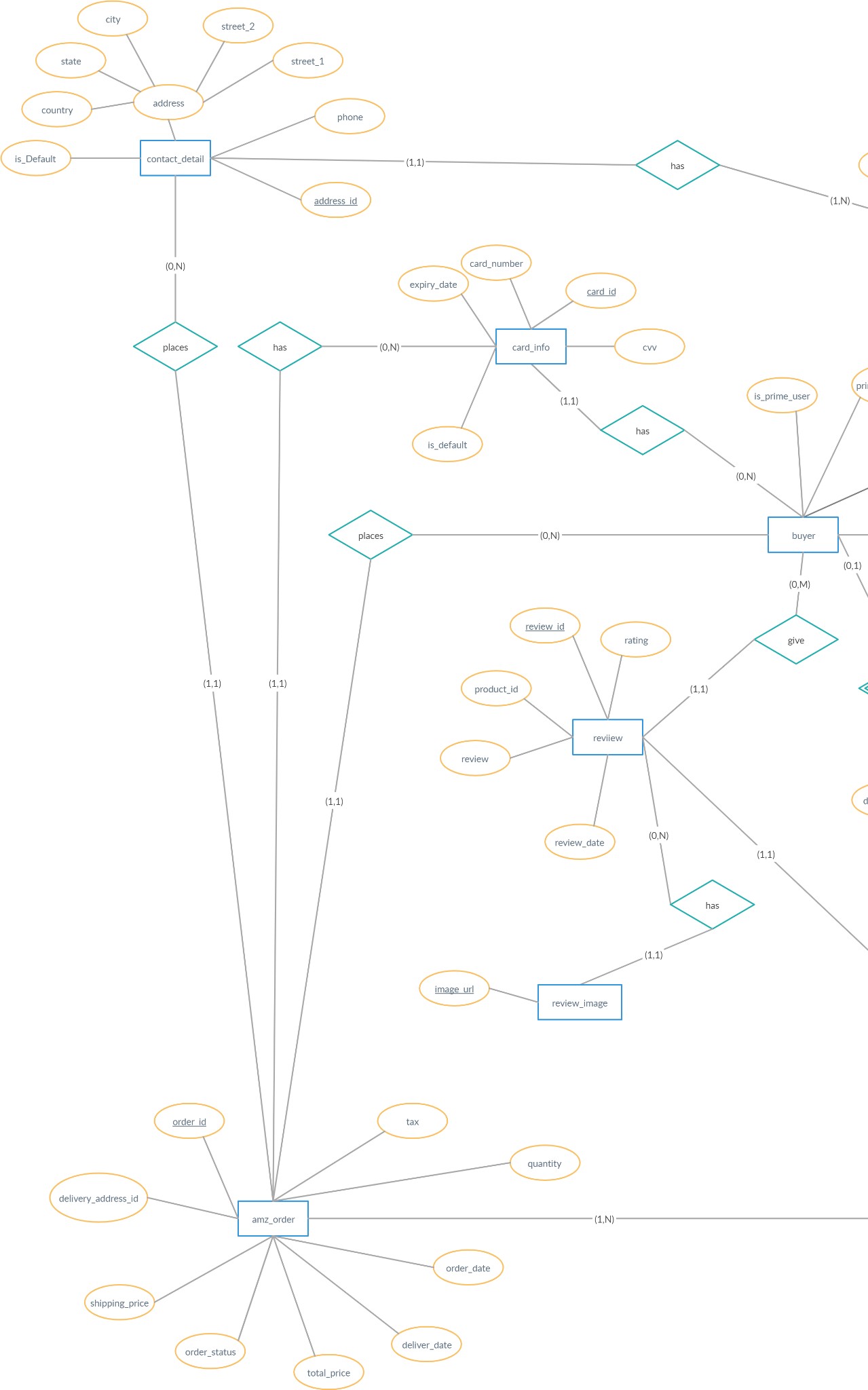
# ENTITY RELATION DIAGRAM

**IMAGE URL:** [https://drive.google.com/drive/folders/1Z5GZC-](https://drive.google.com/drive/folders/1Z5GZC-_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing)

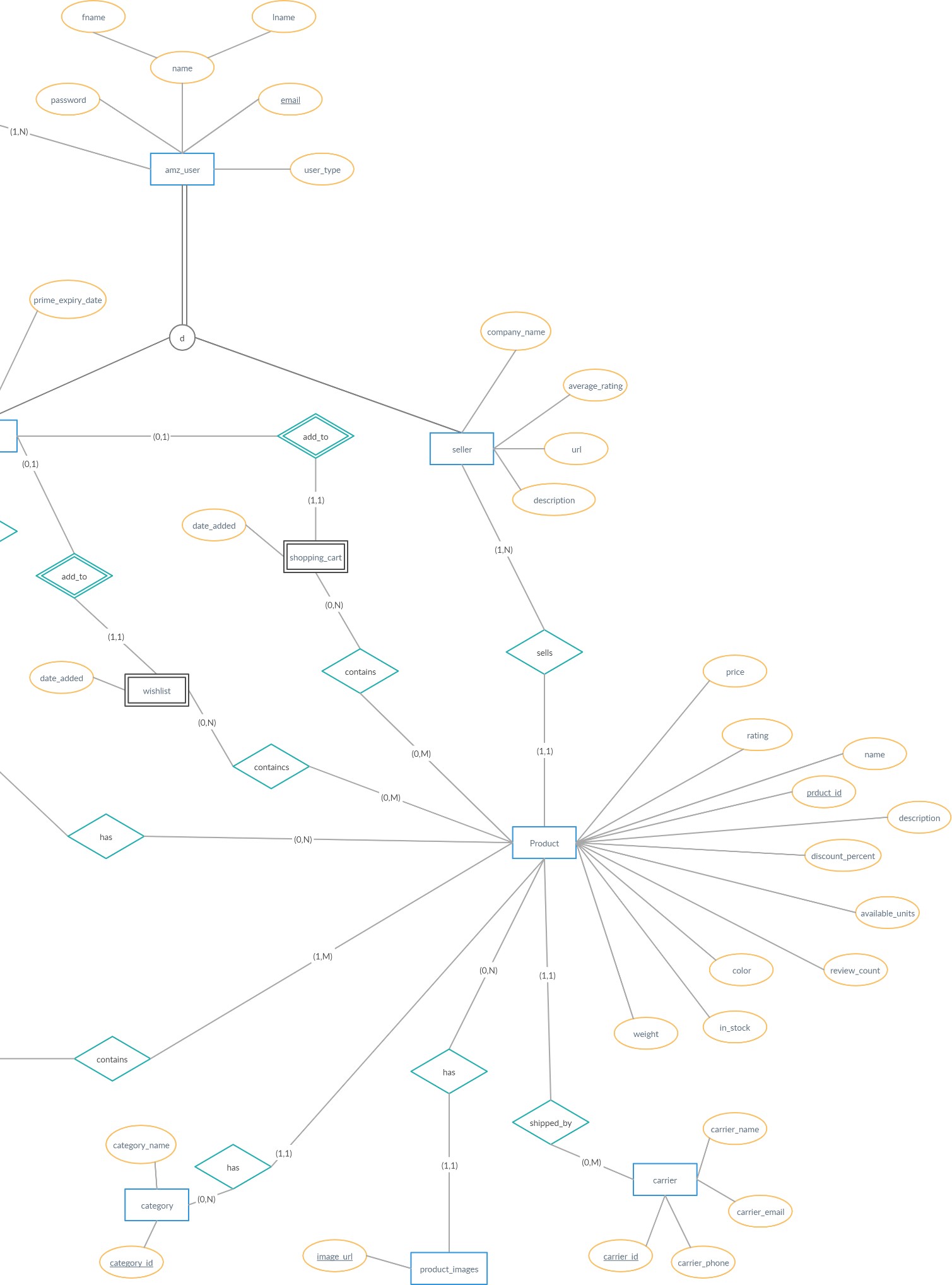
[\_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing](https://drive.google.com/drive/folders/1Z5GZC-_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing)



**Fig 1.0** Amazon’s Entity Relation diagram (big picture)



**Fig 1.1** Amazon’s Entity Relation diagram (left portion)



**Fig 1.2** Amazon’s Entity Relation diagram (right portion)

# RELATIONAL SCHEMA

To map ER diagram into a relational schema, we considered the following mapping rules.

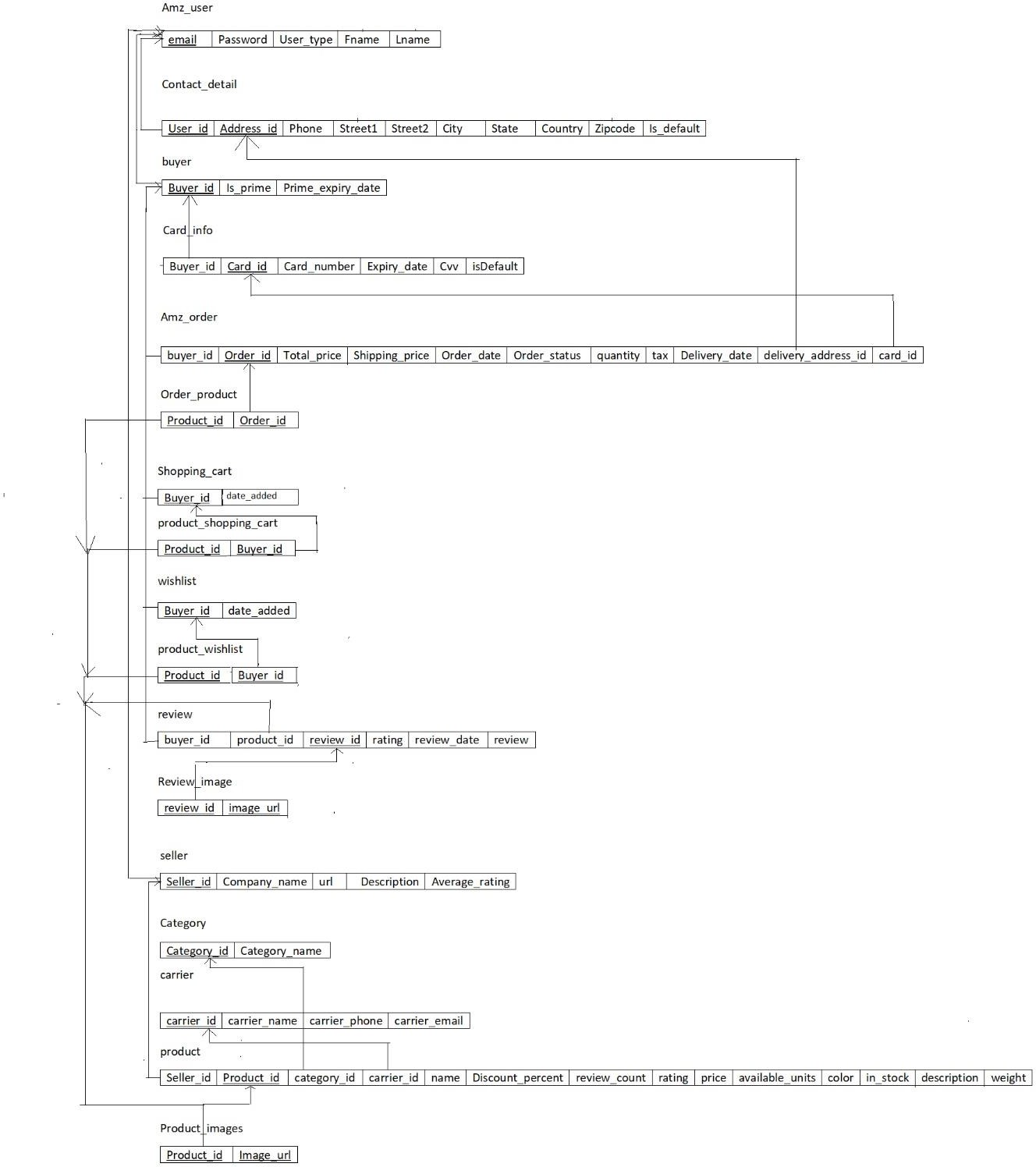
For each 1: 1 binary relationship, in the total participation entity add the primary key of the other entity as the foreign key.

For 1: N binary relationship, add to the entity on the N side the primary key of the other entity as the foreign key.

For M: N binary relationship, make a new entity with foreign key as the primary key of the two participating entities. Their combination forms the new primary key.

* In buyer table we have user\_id as foreign key.
* In seller table we have user\_id as foreign key.
* In product table we have seller\_id, carrier\_id and category\_id as foreign keys.
* In order table we have buyer\_id as foreign key.
* In card\_info we have buyer\_id as foreign key.
* In reviews we have buyer\_id, product\_id as foreign keys.
* In wishlist we have buyer\_id as foreign key.
* In shopping cart we have buyer\_id as foreign key.
* In contact\_details we have user\_id as foreign key.
* In review\_images we have review\_id as foreign key.
* In product\_images we have product\_id as foreign key.
* We make a new table name product\_wishlist which as product\_id and wishlist\_id as foreign key.
* We make a new table name product\_shopping\_cart which as product\_id and buyer\_id as foreign key.
* We make a new table name product\_order which as product\_id and order\_id as foreign key.

After converting the ER Model of our system into relational tables by following the strict guidelines of mapping, we analysed and confirmed that **the resultant tables do not violate any conditions of 3NF normal form**. Thus, the resultant relational tables formed are already in a 3NF normalised form.



**Fig 2** Relational Tables

# SIGNIFICANT PROCEDURES

1. **Register buyer:** invoked by buyer responsible for
   1. registering user given email, fname, lname and password.
   2. registering the buyer itself setting its prime membership as false by default and prime member expiry date as Null.
2. **Register seller** has 2 responsibilities
   1. registering user given email, fname, lname and password.
   2. registering the seller itself given company name, url, description, setting its average rating as

2.5 default.

1. **Place order:** Given a buyer id, place order is responsible for the following –
   1. To iterate over the shopping cart of a particular buyer and remove each item from buyer’s cart.
   2. While removing each product, sum up the price of each product to the total price for the order.
   3. If the user is **“prime user”** do not include shipping charges for that order.
   4. Fetch the default address set by the buyer from the list of addresses for that buyer from the contact\_details tables.
   5. Fetch the default card details set by the buyer from the list of card details for that buyer from the card\_info table.
   6. To make sure not to include certain products from the shopping car in the order table who’s

available unity is zero (in\_stock bit is set to 0).

* 1. To add an entry in the order table, containing details of the invoice (total price, total quantity of products in order, tax, shipping charge, card\_details used for the order, delivery contact details used for the order)
  2. To invokes a trigger responsible for updating the available\_units of each product being bought in that order.

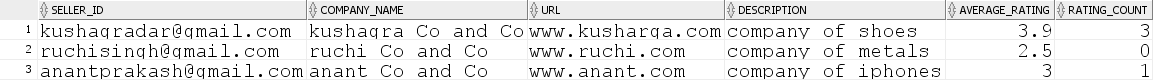
1. **Give review:** A buyer can add a review by providing product\_id, buyer\_id, review, rating, and image\_url (if any). It adds a review in the review table and image of it in the image table. After execution It invokes two triggers.
   1. update\_product\_rating
   2. updte\_seller\_rating.
2. **Add\_contact\_details:** adds details about address and the user’s number. A user can add multiple contact details and can set a single to contact\_details to be used by default.
3. **Add\_card\_info:** add cards information like card number, expiry\_date etc. A user can add mupltiple card details and can set a single to card\_info to be used by default.
4. **Add\_prodcut:** adds a product sold a by a seller. It also asks for the image URL if any, A seller can upload multiple images for a product.
5. **Add\_to\_shopping\_cart:** adds a product in the shopping cart for a buyer, given the buyer Id
6. **Add\_to\_wishlist:** adds a product to the wishlist of a buyer, given the buyer id.
7. **Update\_membership:** update users prime membership information.
8. **Cancel\_membership:** cancel user’s prime membership
9. **Populate\_product\_categories:** adds all the available categories of product in it
10. **Populate\_carrier\_categories:** adds all the available carrier serviced responsible for delivering products.

# IMPORTANT TRIGGERS

1. **Update available units:**
   * Trigger is invoked whenever a user places an order. “**Update available units”** is responsible for updating the value of available units for each product in the product table that is being ordered by a buyer.
   * The procedures iterate over all the entries of order\_product table for a specific order and iteratively updates each products quantity in the product table.
   * If the quantity reaches 0, the product is marked as out of stock, setting its in\_stock value as 0.
2. **Update\_product\_rating:**
   * It is responsible for updating the rating of a product every time a buyer gives a review by averaging the earlier rating with this buyers rating.
   * It also updates the count of rating given for that particular product.
3. **Update\_seller\_rating**
   * It is responsible for updating the rating of a seller every time a buyer gives a review to a product by averaging the earlier rating with this buyers rating.
4. **Remove\_products\_from\_cart**
   * After placing an order by a buyer, the trigger is responsible for removing all the ordered products from the shopping cart of that particular buyer.

# PROJECT IMPLEMTATION AND RESULTS

1. seller table registering the seller,



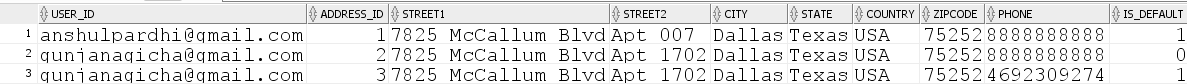
1. buyer table registering the buyer



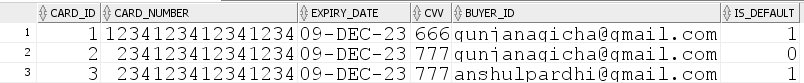
1. updating the buyer’s membership as prime user



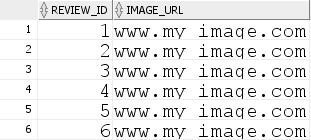
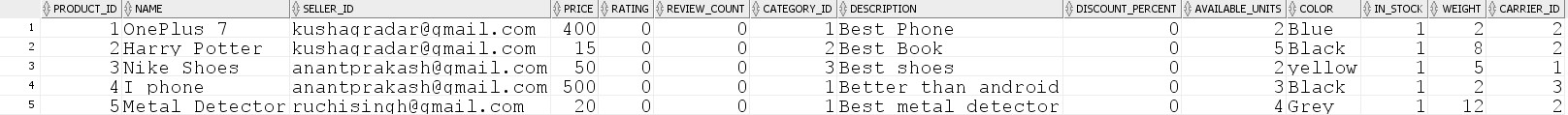
1. contact\_info table after adding contact details by a buyer



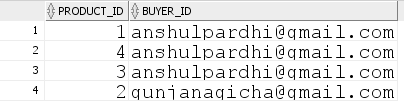
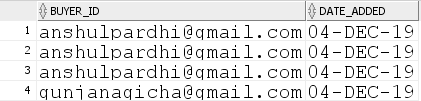
1. card\_info table after adding card details for the payment by a buyer



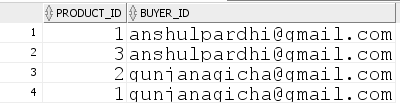
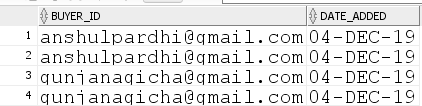
1. product, product image table after uploading products by a seller on the amazon db



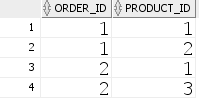
1. wishlist, product\_wishlist table after adding items to wishlist by a user



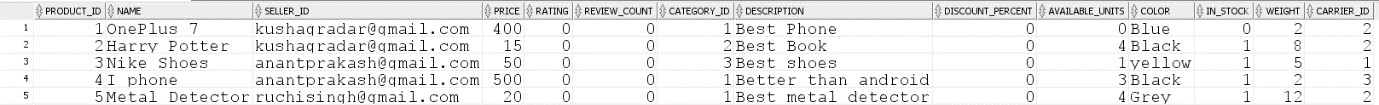
1. shopping cart, product\_shopping table after adding products to shopping cart by a buyer



1. Multiple resultant tables after placing the order,
   * Order table, order product tables (shipping charge is set to zero if user is prime)



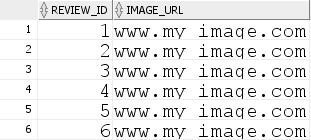
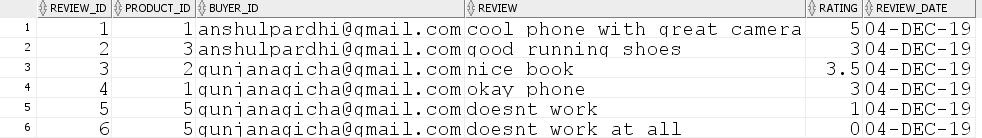
* + Reduce the product’s quantity. (if reached zero, set as out of stock)



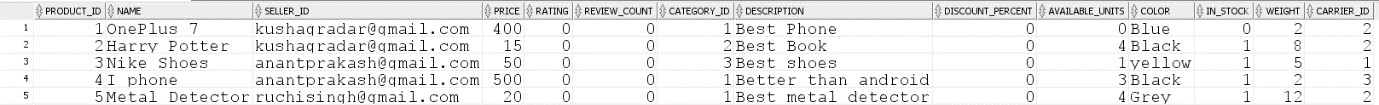
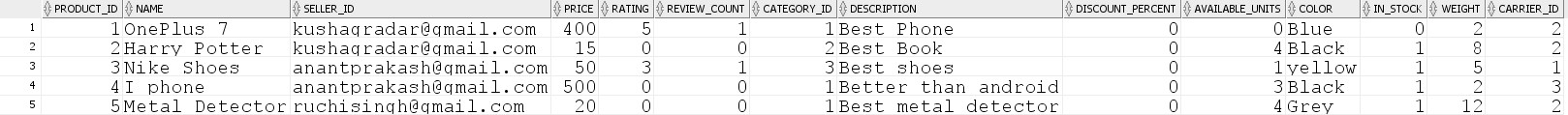
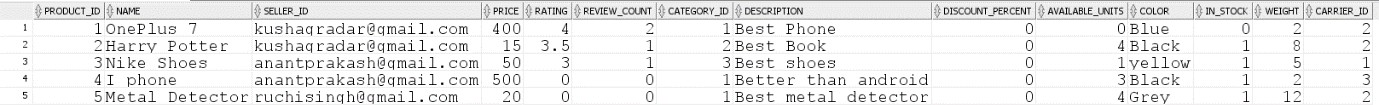
* + Results of trigger after removing bought products from the shopping cart (empty cart)

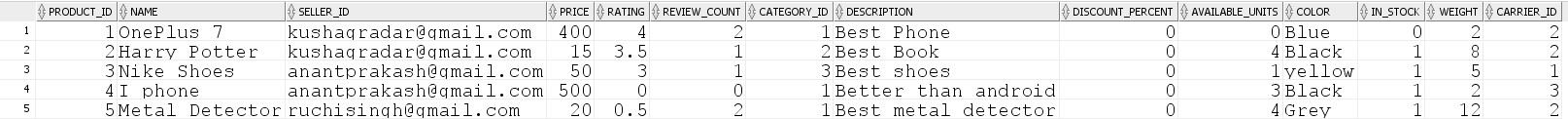


1. Multiple tables are updated after giving a review
   * Review table, product\_review, review image

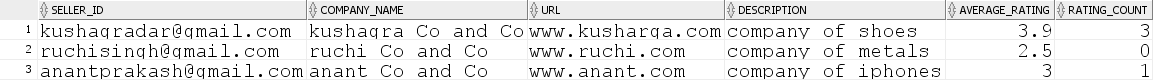


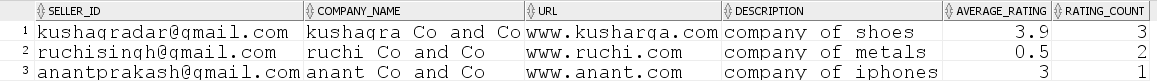
* + Updated product rating based on the given review





* + Updated seller rating based on review





# SQL CODE FOR THE AMAZON DB SYSTEM

**SOURCE CODE:** [https://drive.google.com/drive/folders/1Z5GZC-](https://drive.google.com/drive/folders/1Z5GZC-_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing)

[\_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing](https://drive.google.com/drive/folders/1Z5GZC-_1xq8R5TdcaSBOElE96McfDIa7?usp=sharing)

# Code to create tables and apply contraints

CREATE TABLE amz\_user (

email VARCHAR(255) PRIMARY KEY, fname VARCHAR(255) NOT NULL, lname VARCHAR(255),

password VARCHAR(30) NOT NULL,

user\_type NUMBER(1) NOT NULL

);

CREATE TABLE contact\_detail (

user\_id VARCHAR(255) NOT NULL,

address\_id INTEGER PRIMARY KEY, street1 VARCHAR(255) NOT NULL, street2 VARCHAR(255),

city VARCHAR(50) NOT NULL, state VARCHAR(50) NOT NULL, country VARCHAR(50) NOT NULL, zipcode NUMBER(5) NOT NULL, phone VARCHAR(20) NOT NULL,

is\_default NUMBER(1) DEFAULT 0

);

CREATE TABLE card\_info (

card\_id INTEGER PRIMARY KEY,

card\_number NUMBER(16) NOT NULL, expiry\_date DATE NOT NULL,

cvv NUMBER(3) NOT NULL,

buyer\_id VARCHAR(255) NOT NULL,

is\_default NUMBER(1)

);

CREATE TABLE buyer (

buyer\_id VARCHAR(255) PRIMARY KEY,

is\_prime NUMBER(1) DEFAULT 0, prime\_expiry\_date DATE

);

CREATE TABLE seller (

seller\_id VARCHAR(255) PRIMARY KEY,

company\_name VARCHAR(255) NOT NULL, url VARCHAR(255),

description VARCHAR(255), average\_rating NUMBER(2, 1) DEFAULT 2.5, rating\_count NUMBER DEFAULT 0

);

CREATE TABLE category (

category\_id INTEGER PRIMARY KEY, category\_name VARCHAR(255) NOT NULL

);

CREATE TABLE product (

product\_id INTEGER PRIMARY KEY, name VARCHAR(255) NOT NULL,

seller\_id VARCHAR(255) NOT NULL, price NUMBER(10, 2) NOT NULL,

rating NUMBER(2, 1), review\_count INTEGER, category\_id INTEGER, description VARCHAR(255), discount\_percent NUMBER(4, 2), available\_units INTEGER,

color VARCHAR(30),

in\_stock NUMBER(1),

weight NUMBER(10, 2),

carrier\_id INTEGER

);

CREATE TABLE product\_image ( product\_id INTEGER, image\_url VARCHAR(255),

PRIMARY KEY ( product\_id, image\_url )

);

CREATE TABLE shopping\_cart ( buyer\_id VARCHAR(255), date\_added DATE

);

CREATE TABLE product\_shoppingcart ( product\_id INTEGER,

buyer\_id VARCHAR(255), PRIMARY KEY ( product\_id, buyer\_id )

);

CREATE TABLE wish\_list ( buyer\_id VARCHAR(255), date\_added DATE

);

CREATE TABLE product\_wishlist ( product\_id INTEGER, buyer\_id VARCHAR(255), PRIMARY KEY ( product\_id,

buyer\_id )

);

CREATE TABLE amz\_order (

order\_id INTEGER PRIMARY KEY, buyer\_id VARCHAR(255) NOT NULL,

card\_id INTEGER NOT NULL, total\_price NUMBER(10, 2), order\_date DATE,

tax NUMBER(4, 2) DEFAULT 10,

shipping\_price NUMBER(4, 2) DEFAULT 10, delivery\_address\_id INTEGER,

delivery\_date DATE,

order\_status CHAR(1) NOT NULL, quantity INTEGER NOT NULL

);

CREATE TABLE order\_product ( order\_id INTEGER,

product\_id INTEGER, PRIMARY KEY ( order\_id,

product\_id )

);

CREATE TABLE review (

review\_id INTEGER PRIMARY KEY, product\_id INTEGER NOT NULL, buyer\_id VARCHAR(255) NOT NULL, review VARCHAR(1000),

rating NUMBER(2, 1), review\_date DATE

);

CREATE TABLE review\_image ( review\_id INTEGER, image\_url VARCHAR(255), PRIMARY KEY ( review\_id,

image\_url )

);

CREATE TABLE carrier (

carrier\_id INTEGER PRIMARY KEY, carrier\_name VARCHAR(255) NOT NULL, carrier\_phone NUMBER(10) NOT NULL, carrier\_email VARCHAR(255) NOT NULL

);

ALTER TABLE contact\_detail

ADD CONSTRAINT contact\_detail\_user\_id\_fk FOREIGN KEY ( user\_id ) REFERENCES amz\_user ( email )

## ON DELETE CASCADE;

ALTER TABLE card\_info

ADD CONSTRAINT card\_info\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE product

ADD CONSTRAINT product\_seller\_id\_fk FOREIGN KEY ( seller\_id ) REFERENCES seller ( seller\_id )

## ON DELETE CASCADE;

ALTER TABLE product

ADD CONSTRAINT product\_category\_id\_fk FOREIGN KEY ( category\_id ) REFERENCES category ( category\_id )

## ON DELETE CASCADE;

ALTER TABLE product

ADD CONSTRAINT product\_carrier\_id\_fk FOREIGN KEY ( carrier\_id ) REFERENCES carrier ( carrier\_id )

## ON DELETE CASCADE;

ALTER TABLE product\_image

ADD CONSTRAINT product\_image\_product\_id\_fk FOREIGN KEY ( product\_id ) REFERENCES product ( product\_id )

## ON DELETE CASCADE;

ALTER TABLE shopping\_cart

ADD CONSTRAINT shopping\_cart\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE product\_shoppingcart

ADD CONSTRAINT product\_sc\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE product\_shoppingcart

ADD CONSTRAINT product\_sc\_product\_id\_fk FOREIGN KEY ( product\_id ) REFERENCES product ( product\_id )

## ON DELETE CASCADE;

ALTER TABLE wish\_list

ADD CONSTRAINT wishlist\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE product\_wishlist

ADD CONSTRAINT product\_wishlist\_product\_id\_fk FOREIGN KEY ( product\_id ) REFERENCES product ( product\_id )

## ON DELETE CASCADE;

ALTER TABLE product\_wishlist

ADD CONSTRAINT product\_wishlist\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE amz\_order

ADD CONSTRAINT order\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE amz\_order

ADD CONSTRAINT order\_card\_id\_fk FOREIGN KEY ( card\_id ) REFERENCES card\_info ( card\_id )

## ON DELETE CASCADE;

ALTER TABLE amz\_order

ADD CONSTRAINT order\_delivery\_address\_id\_fk FOREIGN KEY ( delivery\_address\_id ) REFERENCES contact\_detail ( address\_id )

## ON DELETE CASCADE;

ALTER TABLE order\_product

ADD CONSTRAINT order\_product\_order\_id\_fk FOREIGN KEY ( order\_id ) REFERENCES amz\_order ( order\_id )

## ON DELETE CASCADE;

ALTER TABLE order\_product

ADD CONSTRAINT order\_product\_product\_id\_fk FOREIGN KEY ( product\_id ) REFERENCES product ( product\_id )

## ON DELETE CASCADE;

ALTER TABLE review

ADD CONSTRAINT review\_product\_id\_fk FOREIGN KEY ( product\_id ) REFERENCES product ( product\_id )

## ON DELETE CASCADE;

ALTER TABLE review

ADD CONSTRAINT review\_buyer\_id\_fk FOREIGN KEY ( buyer\_id ) REFERENCES buyer ( buyer\_id )

## ON DELETE CASCADE;

ALTER TABLE review\_image

ADD CONSTRAINT review\_image\_review\_id\_fk FOREIGN KEY ( review\_id ) REFERENCES review ( review\_id )

ON DELETE CASCADE;

# Code for stored procedure and triggers

CREATE OR REPLACE PROCEDURE register\_buyer (

email IN VARCHAR, fname IN VARCHAR, lname IN VARCHAR, password IN VARCHAR

## ) AS BEGIN

INSERT INTO amz\_user VALUES ( email,

fname, lname, password, 0

);

INSERT INTO buyer VALUES (

email, 0, NULL

);

END register\_buyer;

CREATE OR REPLACE PROCEDURE register\_seller ( email IN VARCHAR,

fname IN VARCHAR,

lname IN VARCHAR, password IN VARCHAR, company\_name IN VARCHAR, url IN VARCHAR,

description\_var IN VARCHAR

## ) AS BEGIN

INSERT INTO amz\_user VALUES ( email,

fname, lname, password, 1

);

INSERT INTO seller VALUES (

email,

company\_name, url, description\_var, 2.5,

0

);

END register\_seller;

CREATE OR REPLACE PROCEDURE add\_contact\_details ( user\_id IN VARCHAR,

address\_id IN INTEGER, street1 IN VARCHAR, street2 IN VARCHAR, city IN VARCHAR,

state IN VARCHAR, country IN VARCHAR, zipcode IN NUMBER, phone IN VARCHAR

## ) AS BEGIN

INSERT INTO contact\_detail VALUES ( user\_id,

address\_id, street1, street2, city,

state, country, zipcode, phone, 0

);

END add\_contact\_details;

CREATE OR REPLACE PROCEDURE set\_default\_contact\_details ( contact\_id IN INTEGER,

buyer\_id IN VARCHAR

## ) AS

BEGIN

UPDATE contact\_detail SET

is\_default = 1 WHERE

user\_id = buyer\_id

AND address\_id = contact\_id; END set\_default\_contact\_details;

CREATE OR REPLACE PROCEDURE add\_card\_info ( buyer\_id IN VARCHAR,

card\_id IN INTEGER, card\_number IN NUMBER, expiry\_date IN DATE,

cvv IN NUMBER

## ) AS BEGIN

INSERT INTO card\_info VALUES ( card\_id,

card\_number, expiry\_date, cvv,

buyer\_id, 0

);

END add\_card\_info;

CREATE OR REPLACE PROCEDURE set\_default\_card\_info ( card\_id\_var IN INTEGER,

buyer\_id\_var IN VARCHAR

## ) AS BEGIN

UPDATE card\_info SET

is\_default = 1 WHERE

buyer\_id = buyer\_id\_var AND card\_id = card\_id\_var;

END set\_default\_card\_info;

CREATE OR REPLACE PROCEDURE add\_product ( product\_id IN INTEGER,

name IN VARCHAR,

seller\_id IN VARCHAR, price IN NUMBER, category\_id IN INTEGER, description IN VARCHAR, available\_units IN INTEGER, color IN VARCHAR,

weight IN NUMBER, carrier\_id IN INTEGER, image\_url IN VARCHAR

## ) AS BEGIN

INSERT INTO product VALUES (

product\_id, name, seller\_id, price,

0,

0,

category\_id, description, 0,

available\_units, color,

1,

weight, carrier\_id

);

INSERT INTO product\_image VALUES ( product\_id,

image\_url

);

END add\_product;

CREATE OR REPLACE PROCEDURE add\_to\_shopping\_cart ( buyer\_id IN VARCHAR,

product\_id IN INTEGER

## ) AS BEGIN

INSERT INTO shopping\_cart VALUES ( buyer\_id,

sysdate

);

INSERT INTO product\_shoppingcart VALUES ( product\_id,

buyer\_id

);

END add\_to\_shopping\_cart;

CREATE OR REPLACE PROCEDURE add\_to\_wish\_list ( buyer\_id IN VARCHAR,

product\_id IN INTEGER

## ) AS BEGIN

INSERT INTO wish\_list VALUES ( buyer\_id,

sysdate

);

INSERT INTO product\_wishlist VALUES ( product\_id,

buyer\_id

);

END add\_to\_wish\_list;

CREATE OR REPLACE PROCEDURE give\_review ( review\_id IN NUMBER,

product\_id IN INTEGER, buyer\_id IN VARCHAR, review IN VARCHAR,

rating IN NUMBER, image\_url IN VARCHAR

## ) AS BEGIN

INSERT INTO review VALUES (

review\_id, product\_id, buyer\_id, review, rating, sysdate

);

INSERT INTO review\_image VALUES ( review\_id,

image\_url

);

END give\_review;

CREATE OR REPLACE TRIGGER update\_product\_rating AFTER INSERT ON review

## FOR EACH ROW DECLARE

new\_rating NUMBER(2, 1); review\_count\_old INTEGER;

## BEGIN

SELECT

review\_count

INTO review\_count\_old FROM

product WHERE

product\_id = :new.product\_id;

new\_rating := :new.rating; UPDATE product

## SET

rating = ( ( rating \* review\_count\_old ) + new\_rating ) / ( review\_count\_old + 1 ), review\_count = review\_count\_old + 1

## WHERE

product\_id = :new.product\_id;

## END;

CREATE OR REPLACE TRIGGER update\_seller\_rating AFTER INSERT OR UPDATE OF rating ON review

## FOR EACH ROW DECLARE

new\_rating NUMBER(2, 1); seller\_id\_to\_update VARCHAR(255);

## BEGIN

new\_rating := :new.rating; SELECT

seller\_id

INTO seller\_id\_to\_update FROM

product WHERE

product\_id = :new.product\_id;

UPDATE seller SET

average\_rating = ( ( average\_rating \* rating\_count ) + new\_rating ) / ( rating\_count + 1 ), rating\_count = rating\_count + 1

## WHERE

seller\_id = seller\_id\_to\_update; END;

CREATE OR REPLACE PROCEDURE update\_membership ( buyer\_id\_input IN VARCHAR

## ) AS BEGIN

UPDATE buyer SET

is\_prime = 1,

prime\_expiry\_date = add\_months(DATE '2019-11-28', 12) WHERE

buyer\_id = buyer\_id\_input; END update\_membership;

CREATE OR REPLACE PROCEDURE cancel\_membership ( buyer\_id\_input IN VARCHAR

## ) AS BEGIN

UPDATE buyer SET

is\_prime = 0, prime\_expiry\_date = NULL

## WHERE

buyer\_id = buyer\_id\_input; END cancel\_membership;

CREATE OR REPLACE PROCEDURE place\_order ( order\_id IN INTEGER,

buyer\_id\_var IN VARCHAR

## ) AS

card\_id\_var INTEGER; address\_id\_var INTEGER; total\_price\_var NUMBER := 0; curr\_price\_var NUMBER; total\_qty\_var NUMBER := 0; available\_units\_var NUMBER(1); shipping\_price\_var NUMBER := 10; is\_prime\_var NUMBER := 0; CURSOR products\_cur IS

## SELECT

product\_id FROM

product\_shoppingcart WHERE

buyer\_id = buyer\_id\_var;

product\_id\_var INTEGER; BEGIN

OPEN products\_cur; LOOP

FETCH products\_cur INTO product\_id\_var;

EXIT WHEN products\_cur%notfound; SELECT

price, available\_units

## INTO

curr\_price\_var, available\_units\_var

## FROM

product WHERE

product\_id = product\_id\_var;

IF available\_units\_var > 0 THEN

total\_price\_var := ( total\_price\_var + curr\_price\_var ); total\_qty\_var := total\_qty\_var + 1;

INSERT INTO order\_product VALUES ( order\_id,

product\_id\_var

## ); END IF;

-- DELETE FROM product\_shoppingcart

-- WHERE product\_id = product\_id\_var AND buyer\_id = buyer\_id\_var; END LOOP;

CLOSE products\_cur; SELECT

is\_prime

INTO is\_prime\_var FROM

buyer WHERE

buyer\_id = buyer\_id\_var;

IF is\_prime\_var = 1 THEN shipping\_price\_var := 0;

## END IF; SELECT

card\_id

INTO card\_id\_var FROM

card\_info

## WHERE

buyer\_id = buyer\_id\_var AND is\_default = 1;

## SELECT

address\_id

INTO address\_id\_var FROM

contact\_detail WHERE

user\_id = buyer\_id\_var AND is\_default = 1;

total\_price\_var := total\_price\_var + shipping\_price\_var + 10; INSERT INTO amz\_order VALUES (

order\_id, buyer\_id\_var, card\_id\_var, total\_price\_var, sysdate,

10,

shipping\_price\_var, address\_id\_var,

add\_months(DATE '2019-11-28', 1), 'c',

total\_qty\_var

);

END place\_order;

CREATE OR REPLACE PROCEDURE populate\_product\_categories AS BEGIN

INSERT INTO category VALUES ( 1,

'Electronics'

);

INSERT INTO category VALUES ( 2,

'Books'

);

INSERT INTO category VALUES ( 3,

'Clothing'

);

END populate\_product\_categories;

CREATE OR REPLACE PROCEDURE populate\_carriers AS BEGIN

INSERT INTO carrier VALUES ( 1,

## 'DHL', 1234567890,

'DHL@gmail.com'

);

INSERT INTO carrier VALUES ( 2,

'Fedex', 1234567890,

'Fedex@gmail.com'

);

INSERT INTO carrier VALUES ( 3,

## 'UPS', 1234567890,

'UPS@gmail.com'

);

END populate\_carriers;

CREATE OR REPLACE TRIGGER update\_available\_units AFTER INSERT ON amz\_order

## FOR EACH ROW DECLARE

product\_id\_var INTEGER; available\_units\_var INTEGER; CURSOR products\_cur IS SELECT

product\_id FROM

order\_product WHERE

order\_id = :new.order\_id;

## BEGIN

OPEN products\_cur; LOOP

FETCH products\_cur INTO product\_id\_var; EXIT WHEN products\_cur%notfound; SELECT

available\_units

INTO available\_units\_var FROM

product WHERE

product\_id = product\_id\_var;

IF available\_units\_var >= 2 THEN UPDATE product

## SET

available\_units = available\_units - 1 WHERE

product\_id = product\_id\_var;

ELSIF available\_units\_var = 1 THEN UPDATE product

## SET

available\_units = available\_units - 1, in\_stock = 0

## WHERE

product\_id = product\_id\_var; END IF;

## END LOOP;

CLOSE products\_cur; END;

CREATE OR REPLACE TRIGGER remove\_items\_from\_cart AFTER INSERT ON amz\_order

## FOR EACH ROW

DECLARE BEGIN

DELETE FROM shopping\_cart WHERE

buyer\_id = :new.buyer\_id;

DELETE FROM product\_shoppingcart WHERE

buyer\_id = :new.buyer\_id; END;

## BEGIN

register\_buyer('anshulpardhi@gmail.com', 'anshul', 'pardhi', 'abcd123'); register\_buyer('ashwanikashyap@gmail.com', 'ashwani', 'kashyap', 'abcd123'); register\_buyer('gunjanagicha@gmail.com', 'gunjan', 'agicha', 'abcd123');

## END;

BEGIN

register\_seller('kushagradar@gmail.com', 'kushagra', 'dar', 'abcd123', 'kushagra Co and Co',

'www.kusharga.com', 'company of shoes'); register\_seller('ruchisingh@gmail.com', 'ruchi', 'singh', 'abcd123', 'ruchi Co and Co',

'www.ruchi.com', 'company of metals'); register\_seller('anantprakash@gmail.com', 'anant', 'prakash', 'abcd123', 'anant Co and Co',

'www.anant.com', 'company of iphones');

## END;

BEGIN

update\_contact\_details('anshulpardhi@gmail.com', 1, '7825 McCallum Blvd', 'Apt 007', 'Dallas',

'Texas', 'USA', 75252, 8888888888);

update\_contact\_details('gunjanagicha@gmail.com', 2, '7825 McCallum Blvd', 'Apt 1702', 'Dallas',

'Texas', 'USA', 75252, 8888888888);

update\_contact\_details('gunjanagicha@gmail.com', 3, '7825 McCallum Blvd', 'Apt 1702', 'Dallas',

'Texas', 'USA', 75252, 4692309274);

set\_default\_contact\_details(3, 'gunjanagicha@gmail.com'); set\_default\_contact\_details(1, 'anshulpardhi@gmail.com');

## END;

BEGIN

add\_card\_info('gunjanagicha@gmail.com', 1, 1234123412341234, TO\_DATE('2023-12-09', 'YYYY-MM-DD'),

666);

add\_card\_info('gunjanagicha@gmail.com', 2, 0234123412341234, TO\_DATE('2023-12-09', 'YYYY-MM-DD'),

777);

add\_card\_info('anshulpardhi@gmail.com', 3, 0234123412341234, TO\_DATE('2023-12-09', 'YYYY-MM-DD'),

777);

set\_default\_card\_info(1, 'gunjanagicha@gmail.com'); set\_default\_card\_info(3, 'anshulpardhi@gmail.com');

## END;

BEGIN

populate\_product\_categories(); populate\_carriers();

## END;

BEGIN

add\_product(1, 'OnePlus 7', 'kushagradar@gmail.com', 400, 1,

'Best Phone', 2, 'Blue', 2, 2, 'bit.ly/sfdf4fg');

add\_product(2, 'Harry Potter', 'kushagradar@gmail.com', 15, 2,

'Best Book', 5, 'Black', 8, 2, 'bit.ly/sfdf4fg');

add\_product(3, 'Nike Shoes', 'anantprakash@gmail.com', 50, 3,

'Best shoes', 2, 'yellow', 5, 1, 'bit.ly/sfdf4fg');

add\_product(4, 'I phone', 'anantprakash@gmail.com', 500, 1,

'Better than android', 3, 'Black', 2, 3, 'bit.ly/sfdf4fg');

add\_product(5, 'Metal Detector', 'ruchisingh@gmail.com', 20, 1,

'Best metal detector', 4, 'Grey', 12, 2, 'bit.ly/sfdf4fg');

## END; BEGIN

add\_to\_wish\_list('anshulpardhi@gmail.com', 1);

add\_to\_wish\_list('anshulpardhi@gmail.com', 4);

add\_to\_wish\_list('anshulpardhi@gmail.com', 3);

add\_to\_wish\_list('gunjanagicha@gmail.com', 2); END;

## BEGIN

add\_to\_shopping\_cart('anshulpardhi@gmail.com', 1);

add\_to\_shopping\_cart('anshulpardhi@gmail.com', 3);

add\_to\_shopping\_cart('gunjanagicha@gmail.com', 2);

add\_to\_shopping\_cart('gunjanagicha@gmail.com', 1); END;

## BEGIN

update\_membership('gunjanagicha@gmail.com'); END;

## BEGIN

place\_order(1, 'gunjanagicha@gmail.com'); END;

## BEGIN

place\_order(2, 'anshulpardhi@gmail.com'); END;

## BEGIN

give\_review(1, 1, 'anshulpardhi@gmail.com', 'cool phone with great camera', 5, 'www.my\_image.com');

give\_review(2, 3, 'anshulpardhi@gmail.com', 'good running shoes', 3, 'www.my\_image.com');

## END;

BEGIN

give\_review(3, 2, 'gunjanagicha@gmail.com', 'nice book', 3.5, 'www.my\_image.com');

## END;

BEGIN

give\_review(4, 1, 'gunjanagicha@gmail.com', 'okay phone', 3, 'www.my\_image.com');

## END;

BEGIN

give\_review(5, 5, 'gunjanagicha@gmail.com', 'doesnt work', 1, 'www.my\_image.com');

## END;

BEGIN

give\_review(6, 5, 'gunjanagicha@gmail.com', 'doesnt work at all', 0, 'www.my\_image.com');

END;