CS 6360.001 - PROJECT REPORT

Team 10 : AMAZON - 5

Team Members: -

1. Raghav Murali – RXM190067
2. Venkatesh Sankar – VXS200014
3. Yogesh Bala – YXB200007

Functional Requirements: -

1. An Amazon User can be categorized as a Buyer or a Seller.
2. User is uniquely identified by Email used to login to Amazon.
3. User can have multiple contact details linked to their account.
4. Buyer can a prime or a non-prime user.
5. Buyer can add products to Wishlist and cart.
6. Buyer can select Buy Now option to checkout currently selected product.
7. Buyer can place an order.
8. Buyer can cancel an order which was placed.
9. Buyer can return a product and get a refund.
10. Buyer gets a notification once he places an order or makes any change to the order.
11. Buyer can add a payment method and choose between card and Amazon Pay Wallet.
12. Buyer can apply coupons which get added to Amazon Pay Wallet balance after redemption.
13. Buyer can view the generated invoice after payment is successful with unique payment ID.
14. Buyer can add a review for a product by giving rating and adding comments.
15. Seller can add products to the system which can be viewed by users.
16. A product is listed based on its category like electronics, clothing etc.
17. A product is shipped by different shippers.
18. A product has an overall rating and list of reviews based on feedback by users.
19. User can contact a customer representative for any queries using the helpline.

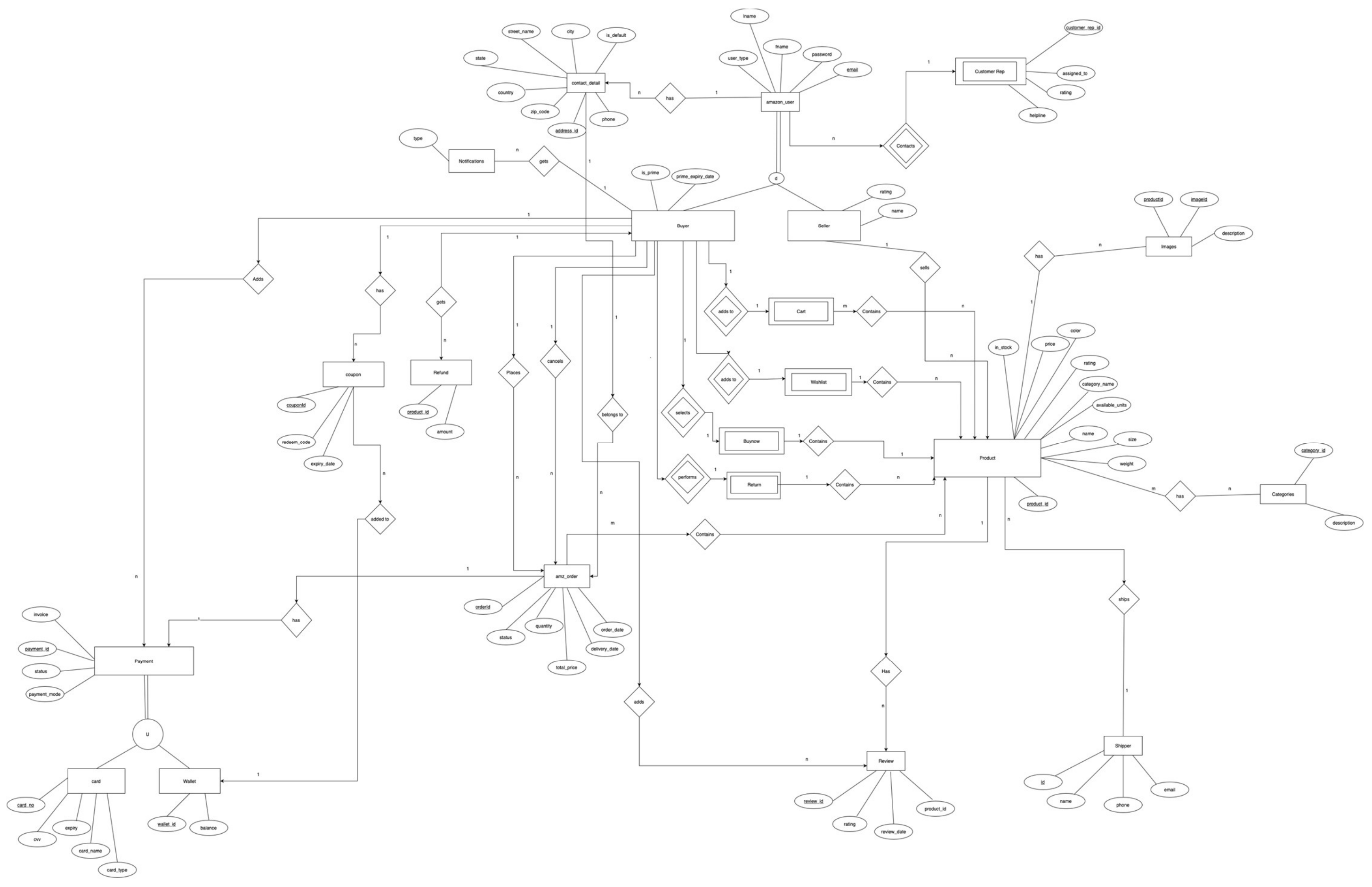
Relationships: -

1. User – Customer representative (1: N) -> Each user can be assigned to one customer representative, and each customer representative can serve multiple customers.
2. User – Contact Details (1: N) -> Each user can have multiple contact details and each contact detail is linked to one user.
3. Buyer – Wishlist (1: 1) -> A buyer can have one Wishlist and each Wishlist is linked to one buyer.
4. Wishlist – Product (M: N) -> A Wishlist can contain multiple products and each product can be part of multiple Wish lists.
5. Buyer – Cart (1: 1) -> A buyer can have one Cart and each Cart is linked to one buyer.
6. Cart – Product (M: N) -> A cart can contain multiple products and each product can be part of multiple carts.
7. Buyer – Buy Now (1: 1) -> A buyer can select Buy now option once at a time and the Buy Now option is linked to one buyer.
8. Buy Now – Product (1: 1) -> A buy now option can be selected for one product at a time and each product can be selected in buy now once at a time.
9. Buyer – Order (1: N) -> A buyer can place multiple orders and each order is placed by a single buyer.
10. Buyer – Order (1: N) -> A buyer can cancel multiple orders and each order is cancelled by one buyer.

11. Buyer – Review (1: N) -> A buyer can add multiple reviews and each review is linked with a single buyer.
12. Buyer – Return (1: 1) -> A buyer can have one return option and each return option is linked to one buyer.
13. Return – Product (1: N) -> A return request can contain multiple products and each product can be returned once.
14. Buyer – Payment (1: N) -> A buyer can have multiple payment modes and each payment mode is linked with one buyer.
15. Buyer – Refund (1: N) -> A buyer can request multiple refunds and each refund is applied to one buyer.
16. Buyer – Coupon (1: N) -> A buyer can have multiple coupons and each coupon is added by one buyer.
17. Buyer – Notification (1: N) -> A buyer gets a notification when any change is done to order placed by the buyer.
18. Coupon – Wallet (M: 1) -> Multiple coupons can be added to one wallet and each wallet can contain multiple coupons.
19. Seller – Product (1: N) -> One seller can put up multiple products for sale. One product can be put up for sale by one seller.
20. Product – Order (M: N) -> Each order can contain multiple products and each product can be part of multiple orders.
21. Product – Review (1: N) -> Each product can have multiple reviews and each review is associated with one product.
22. Product – Shipper (N: 1) -> Each product is shipped by 1 shipper and each shipper can ship multiple products.
23. Order – Payment (1: 1) -> Each order can have one payment and each payment can be linked with one order.
24. Order – Contact Details (N: 1) -> Each order has one contact detail, and each contact detail can be added for multiple orders.

ENTITY – RELATIONSHIP Diagram: -

Please zoom to view the text clearly



Relational Schema: -

We consider the following mapping rules for mapping ER diagram to relational schema.

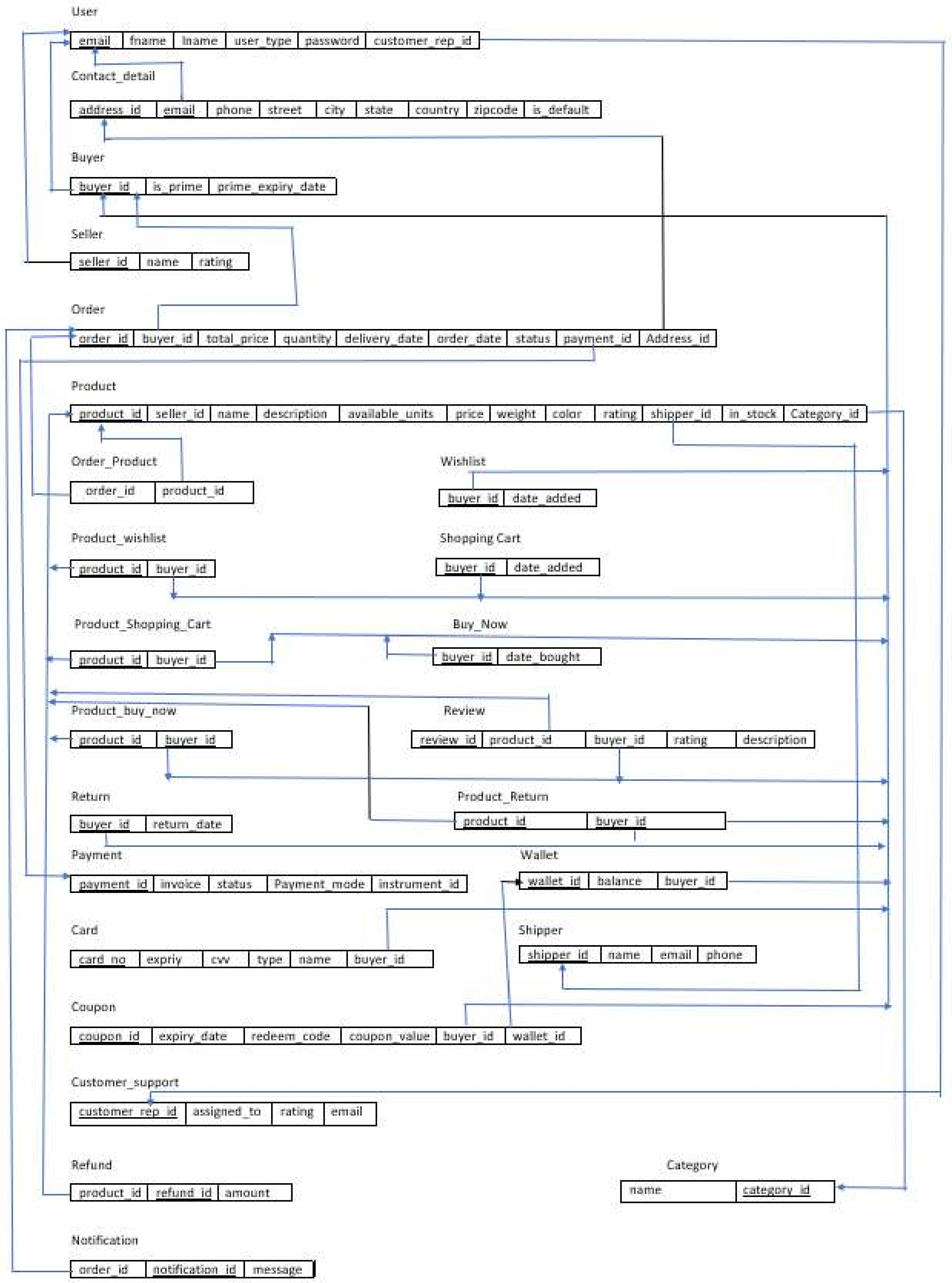
For 1:1 relationship -> In total participation entity, we add the primary key of the other entity as foreign key.

For 1:N relationship -> We add the entity on the N side, the primary key of the other entity as the foreign key.

For M:N relationship -> We make new entity with foreign key as the primary key of the two entities. These two forms the new primary key.

- User table -> customer_rep_id as foreign key
- Contact_detail -> user email as foreign key
- Buyer -> user email as foreign key
- Seller -> user email as foreign key
- Order -> buyer_id , payment_id, address_id as foreign key
- Product -> shipper_id, category_id as foreign key
- Wishlist -> buyer_id as foreign key
- Shopping cart -> buyer_id as foreign key
- Buy_now -> buyer_id as foreign key
- Review -> product _id, buyer_id as foreign key
- Return -> buyer_id as foreign key
- Card -> buyer_id as foreign key
- Coupon -> wallet_id, buyer_id as foreign key
- Refund -> product_id as foreign key
- Notification -> order_id as foreign key
- Product_wishlist -> product_id, buyer_id as foreign key
- Product_shopping_cart -> product_id, buyer_id as foreign key
- Order_product -> order_id, product_id as foreign key

After converting the ER model of Amazon system to relational tables by following the mapping guidelines, we analyzed and confirmed that the resultant tables are already in **third normal form (3NF).**


User
email | fname | lname | user_type | password | customer_rep_id
Contact_detail
address_id | email | phone | street | city | state | country | zipcode | is_default
Buyer
buyer_id | is_prime | prime_expiry_date
Seller
seller_id | name | rating
Order
order_id | buyer_id | total_price | quantity | delivery_date | order_date | status | payment_id | Address_id
Product
product_id | seller_id | name | description | available_units | price | weight | color | rating | shipper_id | in_stock | Category_id
Order_Product
order_id | product_id
Wishlist
buyer_id | date_added
Product_wishlist
product_id | buyer_id
Shopping Cart
buyer_id | date_added
Product_Shopping_Cart
product_id | buyer_id
Buy_Now
buyer_id | date_bought
Product_buy_now
product_id | buyer_id
Review
review_id | product_id | buyer_id | rating | description
Return
buyer_id | return_date
Product_Return
product_id | buyer_id
Payment
payment_id | invoice | status | Payment_mode | instrument_id
Wallet
wallet_id | balance | buyer_id
Card
card_no | expriy | cvv | type | name | buyer_id
Shipper
shipper_id | name | email | phone
Coupon
coupon_id | expiry_date | redeem_code | coupon_value | buyer_id | wallet_id
Customer_support
customer_rep_id | assigned_to | rating | email
Refund
product_id | refund_id | amount
Category
name | category_id
Notification
order_id | notification_id | message

Stored Procedures: -

1. **Add_buyer ->** This procedure is used to register the buyer as an amazon user with fname, lname, email, password and user type. Additional buyer details can also be provided such as prime membership and prime expiry date.

2. **Add_seller ->** This procedure is used to register the seller as an amazon user with fname, lname, email, password and user type. Additional seller details can also be provided such as seller name and default seller rating as 0.

3. **Add_contact_details ->** Allows the user to add contact details such as address, phone and has the option to select a contact details as default which will be used while placing order.

4. **Set_default_contact_detail ->** This procedure is used to set a contact details of a user to default.

5. **Add_shipper ->** Allows the shipper to register themselves into the amazon database with details such as shipper id, phone, shipper name and email.

6. **Update_prime_membership ->** This procedure is used to change the membership of a buyer to a prime membership.

7. **Cancel_prime_membership ->** This procedure is used to cancel the prime membership of a user.

8. **Populate_product_categories ->** Populates the product categories to categorize each product such as Fashion, Electronics and Groceries.

9. **Add_product ->** This procedure is used to add a product into the amazon database with details such as product name, description, available units, product image etc.

10. **Add_to_wishlist ->** Allows the buyer to add a product into his wishlist.

11. **Add_to_shopping_cart ->** Allows the buyer to add a product into his shopping cart.

12. **Add_to_buy_now ->** Allows the buyer to buy a product directly without adding it to the cart.

13. **Add_to_return ->** Allows the buyer to return a product.

14. **Add_review ->** Allows the buyer to add a review about a product with details such as review description and rating. The review is then used in subsequent triggered actions such as updating the product rating.

15. **Add_customer_rep ->** Used to register customer representatives into the amazon database, with default assigned_to as null which indicates if the customer representative is assigned to any amazon user.

16. **Add_card_details ->** Buyer can add his card details into the system with details such as card id, cvv, expiry date, card name and card type. This card can then be used to complete payment while placing an order.

17. **Add_wallet_details ->** This procedure is used to associate a unique wallet with each buyer and has details such as wallet id and balance.

18. **Create_coupon ->** To add a coupon to the amazon database with details such as coupon amount, redeem code and expiry date. The coupon can then be associated with a buyer's profile.

19. **Add_coupon_to_wallet ->** This procedure is used to add the coupon amount to the buyer's wallet and update the wallet balance.

20. **Contact_customer_support ->** This procedure is used to assign a customer support executive to an amazon user in case the executive is not already assigned to another user.

21. **Place_order ->** This procedure is used to place an order and has the following features: -

    a) Fetch all the products from the buyer's shopping cart.
    b) Fetch all the details of the products such as price and available units.
    c) If the available units are more than 0, calculate the total price by adding all the prices of the products in the cart and update the quantity of the order.
    d) If the user is prime user, give a 10% discount on the total order price.
    e) Fetch the card no and wallet id of the user.
    f) Generate the payment id and invoice to be displayed to the user.
    g) Depending on the payment mode specified, generate the payment details.
    h) Fetch the default address to be used for delivering the order.
    i) Once all details are fetched, mark the order as completed and send notification to the user through means of a trigger.

22. **Cancel_order ->** This procedure is used to cancel the order which was placed by the buyer. The order status is set to cancelled and notification is sent to the buyer. The buyer's wallet is then updated with the order amount.

Triggers: -

1. **Remove_items_from_cart ->** This trigger is executed to remove item from a buyer's shopping cart whenever an order is placed successfully.

2. **Update_available_units ->** This trigger is executed to update the available units of a product when it is included in any order and the order is placed successfully. When the available units become 0, update the in-stock flag which indicates that the product is no longer in stock and cannot be added to any order going forward.

3. **Update_product_rating ->** This trigger is executed to update the rating of a product whenever the buyer adds a review and gives the product rating.

4. **Notification_on_update ->** This trigger is executed to send a notification to the buyer whenever any order is placed or cancelled.

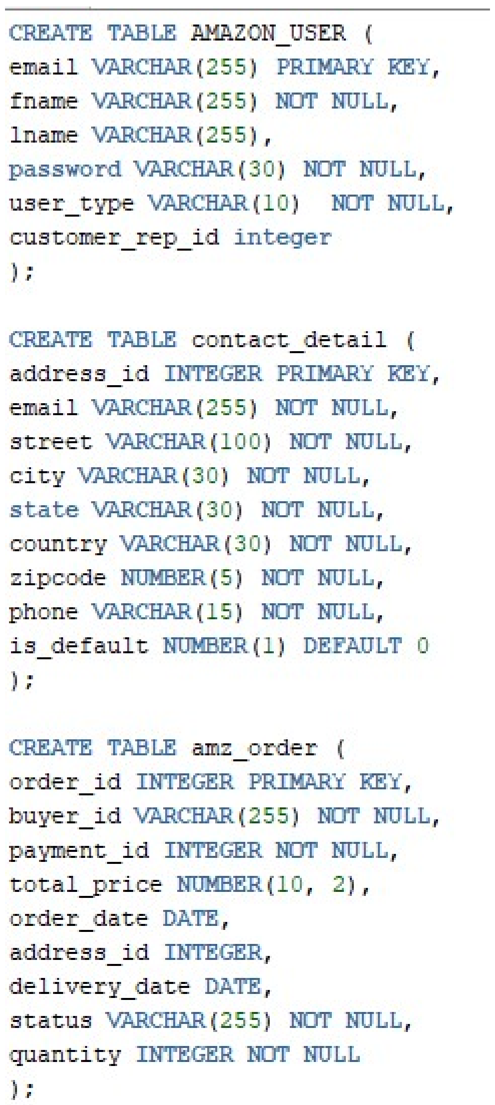
5. **Initiate_refund ->** This trigger is used to initiate refund to the buyer's wallet whenever the buyer returns any product.

Project Implementation and results: -

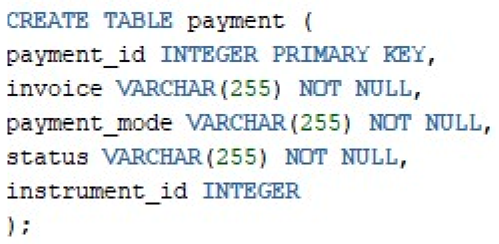
Code can be found in the following Google drive link :-

https://drive.google.com/drive/folders/12bRTc_uZzXgnzKoeiLEZlFaNu3AFAQtY

Creation of tables and constraints: -



```
CREATE TABLE AMAZON_USER (
email VARCHAR(255) PRIMARY KEY,
fname VARCHAR(255) NOT NULL,
lname VARCHAR(255),
password VARCHAR(30) NOT NULL,
user_type VARCHAR(10)  NOT NULL,
customer_rep_id integer
);

CREATE TABLE contact_detail (
address_id INTEGER PRIMARY KEY,
email VARCHAR(255) NOT NULL,
street VARCHAR(100) NOT NULL,
city VARCHAR(30) NOT NULL,
state VARCHAR(30) NOT NULL,
country VARCHAR(30) NOT NULL,
zipcode NUMBER(5) NOT NULL,
phone VARCHAR(15) NOT NULL,
is_default NUMBER(1) DEFAULT 0
);

CREATE TABLE amz_order (
order_id INTEGER PRIMARY KEY,
buyer_id VARCHAR(255) NOT NULL,
payment_id INTEGER NOT NULL,
total_price NUMBER(10, 2),
order_date DATE,
address_id INTEGER,
delivery_date DATE,
status VARCHAR(255) NOT NULL,
quantity INTEGER NOT NULL
);
```



```
CREATE TABLE payment (
payment_id INTEGER PRIMARY KEY,
invoice VARCHAR(255) NOT NULL,
payment_mode VARCHAR(255) NOT NULL,
status VARCHAR(255) NOT NULL,
instrument_id INTEGER
);
```

```sql
CREATE TABLE card (
card_no INTEGER PRIMARY KEY,
expiry DATE,
cvv INTEGER NOT NULL,
card_type VARCHAR(255) NOT NULL,
card_name VARCHAR(255) NOT NULL,
payment_id INTEGER,
buyer_id VARCHAR(255)
);

CREATE TABLE wallet (
wallet_id INTEGER PRIMARY KEY,
balance INTEGER NOT NULL,
payment_id INTEGER,
buyer_id VARCHAR(255)
);

CREATE TABLE buyer (
buyer_id VARCHAR(255) PRIMARY KEY,
is_prime VARCHAR(10) DEFAULT 'no',
prime_expiry_date DATE
);

CREATE TABLE seller (
seller_id VARCHAR(255) PRIMARY KEY,
name VARCHAR(255) NOT NULL,
rating NUMBER(2, 1)
);


CREATE TABLE coupon (
coupon_id INTEGER PRIMARY KEY,
expiry_date DATE,
redeem_code VARCHAR(255) NOT NULL,
buyer_id VARCHAR(255),
wallet_id INTEGER,
coupon_value INTEGER
);


CREATE TABLE product (
product_id INTEGER PRIMARY KEY,
seller_id VARCHAR(255) NOT NULL,
name VARCHAR(255) NOT NULL,
description VARCHAR(200),
available_units INTEGER,
price NUMBER(8, 2) NOT NULL,
weight NUMBER(8, 2),
rating NUMBER(2, 1),
category_id INTEGER,
color VARCHAR(25),
in_stock NUMBER(1),
shipper_id INTEGER
);
```

```sql
CREATE TABLE product_image (
product_id INTEGER,
image_url VARCHAR(255),
PRIMARY KEY ( product_id,image_url )
);


CREATE TABLE review (
review_id INTEGER PRIMARY KEY,
product_id INTEGER,
buyer_id VARCHAR(255),
description VARCHAR(255),
rating INTEGER NOT NULL
);

CREATE TABLE shipper (
shipper_id INTEGER PRIMARY KEY,
phone INTEGER NOT NULL,
name VARCHAR(255) NOT NULL,
email VARCHAR(255) NOT NULL
);

CREATE TABLE customer_support (
customer_rep_id INTEGER PRIMARY KEY,
helpline_no INTEGER NOT NULL,
assigned_to VARCHAR(255),
rating NUMBER(38)
);

CREATE TABLE return (
buyer_id VARCHAR(255),
return_date DATE
);


CREATE TABLE product_return (
buyer_id VARCHAR(255),
product_id INTEGER
);

CREATE TABLE order_product (
order_id INTEGER,
product_id INTEGER,
PRIMARY KEY ( order_id, product_id )
);

CREATE TABLE wishlist (
buyer_id VARCHAR(255),
date_added DATE
);

CREATE TABLE product_wishlist (
product_id INTEGER,
buyer_id VARCHAR(255),
PRIMARY KEY ( product_id, buyer_id )
);
```

```sql
CREATE TABLE shopping_cart (
buyer_id VARCHAR(255),
date_added DATE
);

CREATE TABLE product_shopping_cart (
product_id INTEGER,
buyer_id VARCHAR(255),
PRIMARY KEY ( product_id, buyer_id )
);

CREATE TABLE buy_now (
buyer_id VARCHAR(255),
date_bought DATE
);

CREATE TABLE product_buy_now (
product_id INTEGER,
buyer_id VARCHAR(255),
PRIMARY KEY ( product_id, buyer_id )
);

CREATE TABLE category (
category_id INTEGER PRIMARY KEY,
category_name VARCHAR(255) NOT NULL
);

CREATE TABLE refund (
amount INTEGER,
product_id INTEGER,
refund_id INTEGER PRIMARY KEY
);


CREATE TABLE notification (
order_id INTEGER NOT NULL,
message VARCHAR(255),
notification_id INTEGER
);


ALTER TABLE contact_detail
ADD CONSTRAINT contact_detail_user_email_fk FOREIGN KEY (email)
REFERENCES amazon_user (email)
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;
```

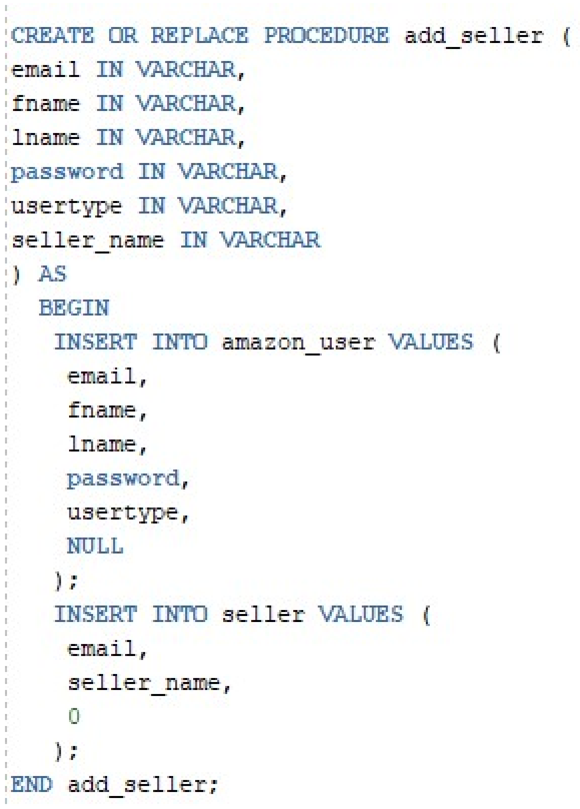
```sql
ALTER TABLE product
ADD CONSTRAINT product_shipper_id_fk FOREIGN KEY ( shipper_id )
REFERENCES shipper (shipper_id)
ON DELETE CASCADE;

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_shopping_cart
ADD CONSTRAINT product_sh_ca_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer (buyer_id)
ON DELETE CASCADE;

ALTER TABLE product_shopping_cart
ADD CONSTRAINT product_sh_ca_product_id_fk FOREIGN KEY ( product_id )
REFERENCES product (product_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 payment_id_fk FOREIGN KEY ( payment_id )
REFERENCES payment ( payment_id )
ON DELETE CASCADE;


ALTER TABLE card
ADD CONSTRAINT card_payment_fk FOREIGN KEY ( payment_id )
REFERENCES payment ( payment_id )
ON DELETE CASCADE;

ALTER TABLE wallet
ADD CONSTRAINT wallet_payment_fk FOREIGN KEY ( payment_id )
REFERENCES payment ( payment_id )
ON DELETE CASCADE;

ALTER TABLE coupon
ADD CONSTRAINT coupon_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer ( buyer_id )
ON DELETE CASCADE;

ALTER TABLE coupon
ADD CONSTRAINT coupon_wallet_id_fk FOREIGN KEY ( wallet_id )
REFERENCES wallet ( wallet_id )
ON DELETE CASCADE;
```

```sql
ALTER TABLE return
ADD CONSTRAINT return_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer ( buyer_id )
ON DELETE CASCADE;

ALTER TABLE product_return
ADD CONSTRAINT product_return_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer ( buyer_id )
ON DELETE CASCADE;

ALTER TABLE product_return
ADD CONSTRAINT product_return_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
ADD CONSTRAINT review_product_id_fk FOREIGN KEY ( product_id )
REFERENCES product ( product_id )
ON DELETE CASCADE;

ALTER TABLE wishlist
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 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;
```

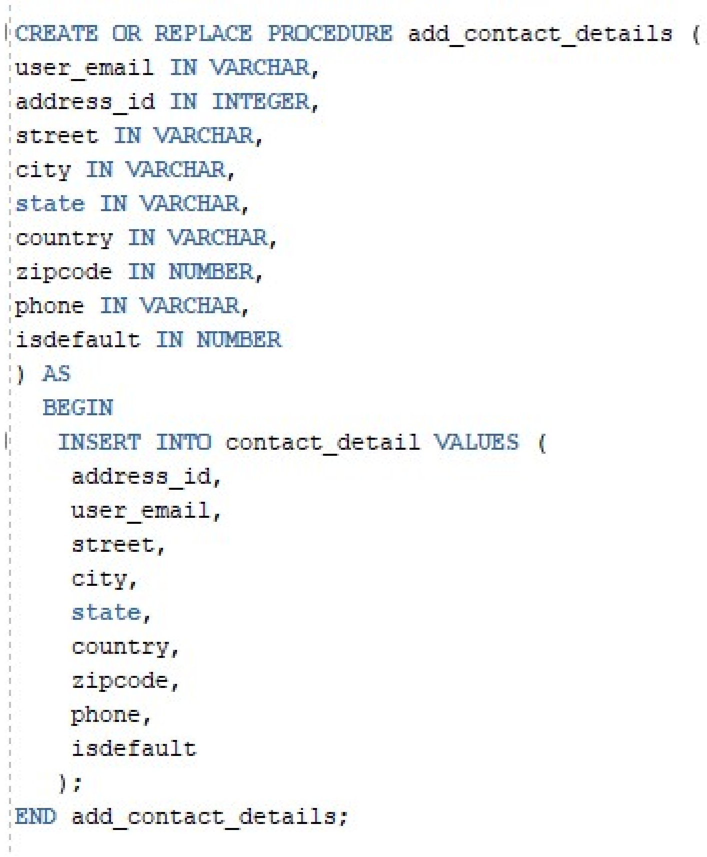
```sql
ALTER TABLE buy_now
ADD CONSTRAINT buy_now_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer (buyer_id)
ON DELETE CASCADE;

ALTER TABLE product_buy_now
ADD CONSTRAINT product_buy_now_buyer_id_fk FOREIGN KEY ( buyer_id )
REFERENCES buyer (buyer_id)
ON DELETE CASCADE;

ALTER TABLE product_buy_now
ADD CONSTRAINT product_buy_now_product_id_fk FOREIGN KEY ( product_id )
REFERENCES product (product_id)
ON DELETE CASCADE;

ALTER TABLE amazon_user
ADD CONSTRAINT user_cust_rep_id_fk FOREIGN KEY (customer_rep_id)
REFERENCES customer_support (customer_rep_id)
ON DELETE CASCADE;

ALTER TABLE refund
ADD CONSTRAINT refund_product_id_fk FOREIGN KEY ( product_id )
REFERENCES product (product_id)
ON DELETE CASCADE;
```

Stored procedures: -

```sql
CREATE OR REPLACE PROCEDURE add_buyer (
email IN VARCHAR,
fname IN VARCHAR,
lname IN VARCHAR,
password IN VARCHAR,
isprime IN VARCHAR,
usertype IN VARCHAR,
prime_expiry_date IN DATE
) AS
   BEGIN
     INSERT INTO amazon_user VALUES (
      email,
      fname,
      lname,
      password,
      usertype,
      NULL
    );
    INSERT INTO buyer VALUES (
     email,
     isprime,
     prime_expiry_date
    );
END add_buyer;
```



```
CREATE OR REPLACE PROCEDURE add_seller (
email IN VARCHAR,
fname IN VARCHAR,
lname IN VARCHAR,
password IN VARCHAR,
usertype IN VARCHAR,
seller_name IN VARCHAR
) AS
  BEGIN
   INSERT INTO amazon_user VALUES (
    email,
    fname,
    lname,
    password,
    usertype,
    NULL
   );
   INSERT INTO seller VALUES (
    email,
    seller_name,
    0
   );
END add_seller;
```

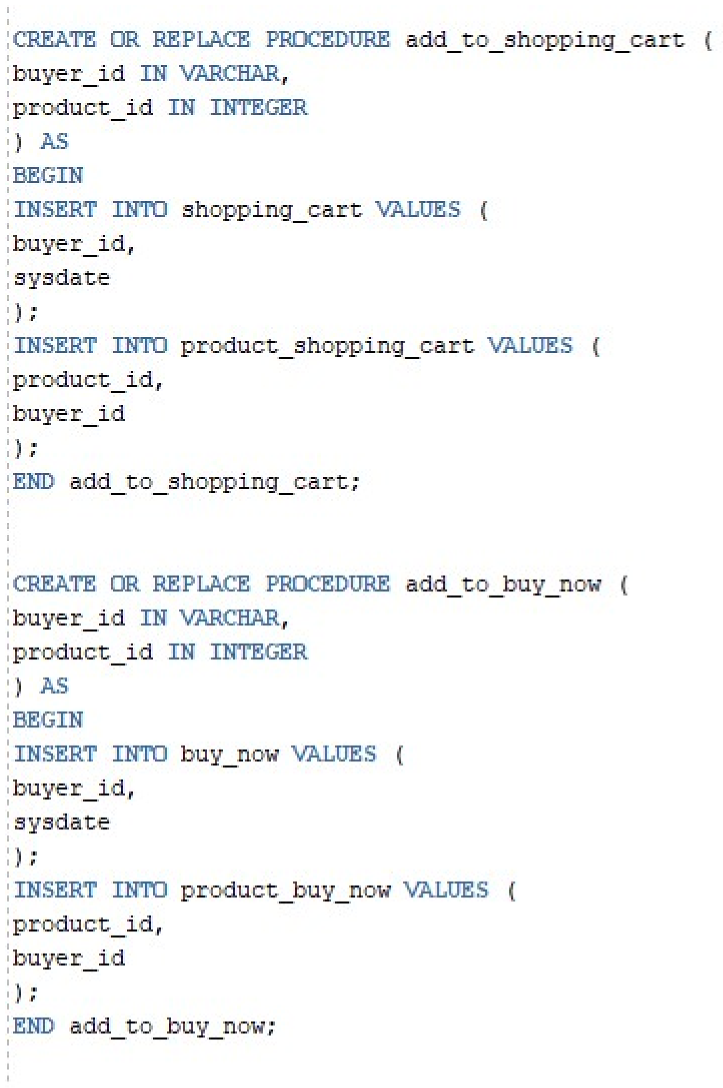


```
CREATE OR REPLACE PROCEDURE add_contact_details (
user_email IN VARCHAR,
address_id IN INTEGER,
street IN VARCHAR,
city IN VARCHAR,
state IN VARCHAR,
country IN VARCHAR,
zipcode IN NUMBER,
phone IN VARCHAR,
isdefault IN NUMBER
) AS
  BEGIN
   INSERT INTO contact_detail VALUES (
    address_id,
    user_email,
    street,
    city,
    state,
    country,
    zipcode,
    phone,
    isdefault
   );
END add_contact_details;
```

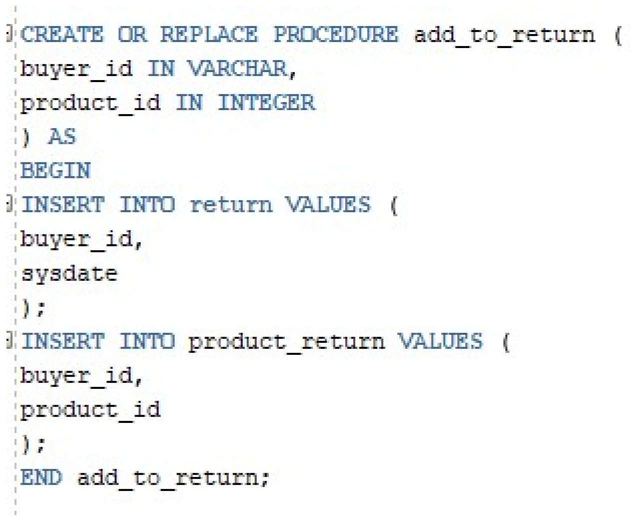
```
CREATE OR REPLACE PROCEDURE populate_product_categories AS
BEGIN
INSERT INTO category VALUES (
1,
'Fashion'
);
INSERT INTO category VALUES (
2,
'Electronics'
);INSERT INTO category VALUES (
3,
'Groceries'
);
END populate_product_categories;
```

```
CREATE OR REPLACE PROCEDURE add_shipper AS
BEGIN
INSERT INTO shipper VALUES (
1,
193842024,
'UPS',
'UPS@gmail.com'
);
INSERT INTO shipper VALUES (
2,
293842024,
'Bluedart',
'Bluedart@gmail.com'
);
INSERT INTO shipper VALUES (
3,
393842026,
'Fedex',
'Fedex@gmail.com'
);
END add_shipper;
```

```
CREATE OR REPLACE PROCEDURE add_product (
product_id IN INTEGER,
seller_id IN VARCHAR,
name IN VARCHAR,
description IN VARCHAR,
available_units IN INTEGER,
price IN NUMBER,
weight IN NUMBER,
category_id IN INTEGER,
color IN VARCHAR,
shipper_id IN INTEGER,
image_url IN VARCHAR
) AS
BEGIN
INSERT INTO product VALUES (
product_id,
seller_id,
name,
description,
available_units,
price,
weight,
0,
category_id,
color,
0,
shipper_id
);
INSERT INTO product_image VALUES (
product_id,
image_url
);
END add_product;

CREATE OR REPLACE PROCEDURE add_to_wishlist (
buyer_id IN VARCHAR,
product_id IN INTEGER
) AS
BEGIN
INSERT INTO wishlist VALUES (
buyer_id,
sysdate
);
INSERT INTO product_wishlist VALUES (
product_id,
buyer_id
);
END add_to_wishlist;
```



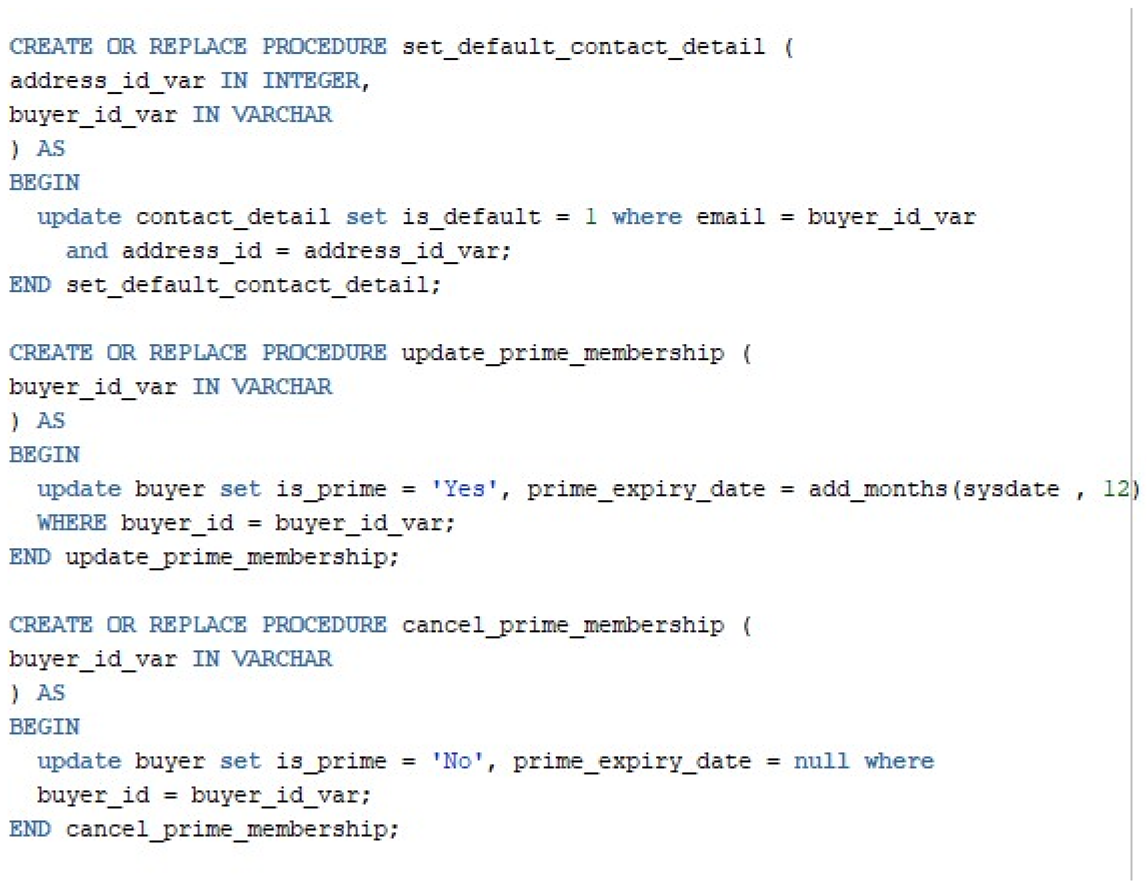
```
CREATE OR REPLACE PROCEDURE add_to_shopping_cart (
buyer_id IN VARCHAR,
product_id IN INTEGER
) AS
BEGIN
INSERT INTO shopping_cart VALUES (
buyer_id,
sysdate
);
INSERT INTO product_shopping_cart VALUES (
product_id,
buyer_id
);
END add_to_shopping_cart;


CREATE OR REPLACE PROCEDURE add_to_buy_now (
buyer_id IN VARCHAR,
product_id IN INTEGER
) AS
BEGIN
INSERT INTO buy_now VALUES (
buyer_id,
sysdate
);
INSERT INTO product_buy_now VALUES (
product_id,
buyer_id
);
END add_to_buy_now;
```



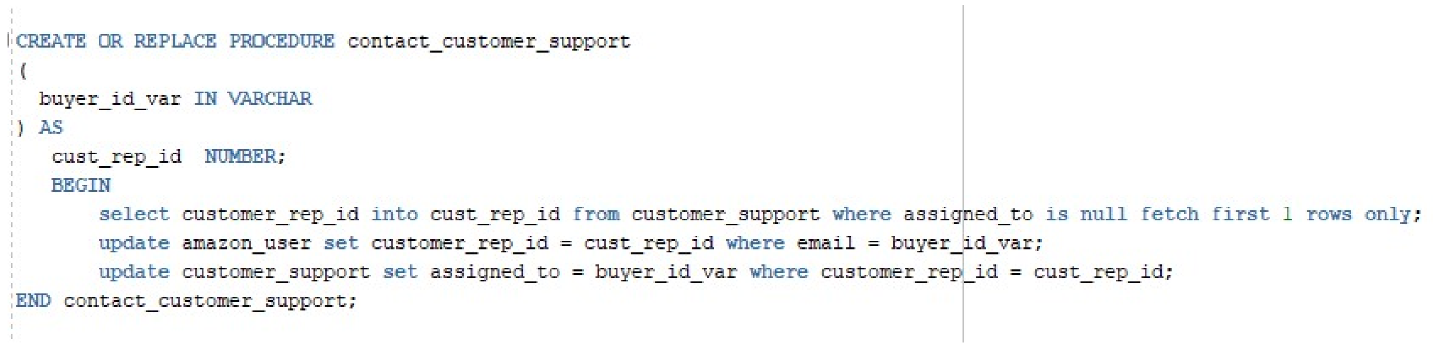
```
CREATE OR REPLACE PROCEDURE add_to_return (
buyer_id IN VARCHAR,
product_id IN INTEGER
) AS
BEGIN
INSERT INTO return VALUES (
buyer_id,
sysdate
);
INSERT INTO product_return VALUES (
buyer_id,
product_id
);
END add_to_return;
```

```
CREATE OR REPLACE PROCEDURE add_review (
review_id IN NUMBER,
product_id IN INTEGER,
buyer_id IN VARCHAR,
description IN VARCHAR,
rating IN NUMBER
) AS
BEGIN
INSERT INTO review VALUES (
review_id,
product_id,
buyer_id,
description,
rating
);
END add_review;

CREATE OR REPLACE PROCEDURE add_customer_rep AS
BEGIN
INSERT INTO customer_support VALUES (
123,
73782973,
null,
0
);
INSERT INTO customer_support VALUES (
456,
93782972,
null,
0
);INSERT INTO customer_support VALUES (
789,
53782923,
null,
0
);
END add_customer_rep;
```

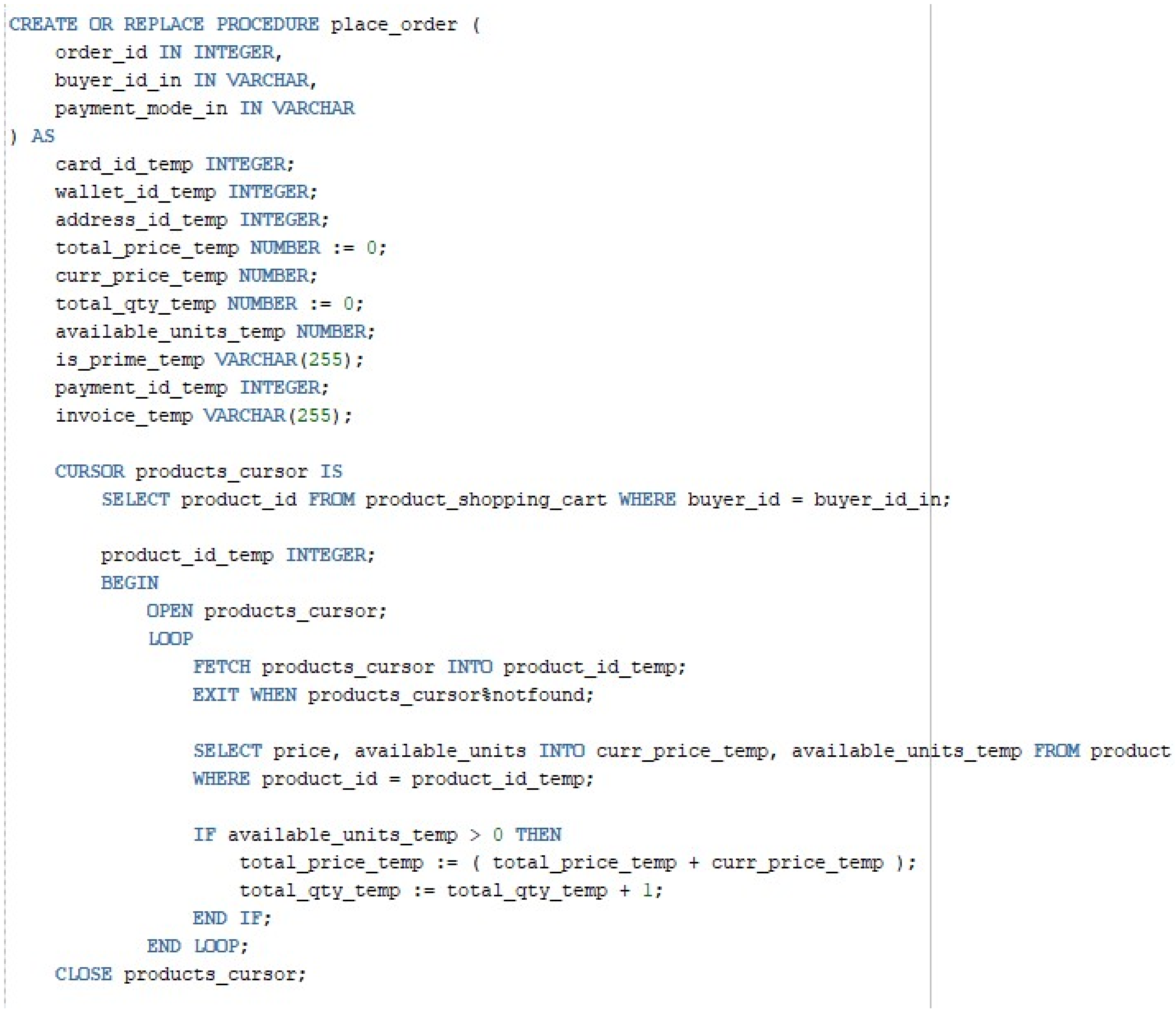
```
CREATE OR REPLACE PROCEDURE add_card_details(
  buyer_id IN VARCHAR,
  card_no IN NUMBER,
  expiry IN DATE,
  cvv IN NUMBER,
  card_type IN VARCHAR,
  card_name IN VARCHAR
) AS
   BEGIN

     INSERT INTO card VALUES (
      card_no,
      expiry,
      cvv,
      card_type,
      card_name,
      buyer_id
      );
END add_card_details;


CREATE OR REPLACE PROCEDURE add_wallet_details(
  buyer_id IN VARCHAR,
  wallet_id IN NUMBER
) AS
   BEGIN

     INSERT INTO wallet VALUES (
      wallet_id,
      0,
      buyer_id
      );
END add_wallet_details;


CREATE OR REPLACE PROCEDURE create_coupon (
  coupon_id IN NUMBER,
  expiry_date IN DATE,
  redeem_code IN VARCHAR,
  coupon_value IN INTEGER
) AS
   BEGIN

     INSERT INTO coupon VALUES (
      coupon_id,
      expiry_date,
      redeem_code,
      null,
      null,
      coupon_value
      );
END create_coupon;
```



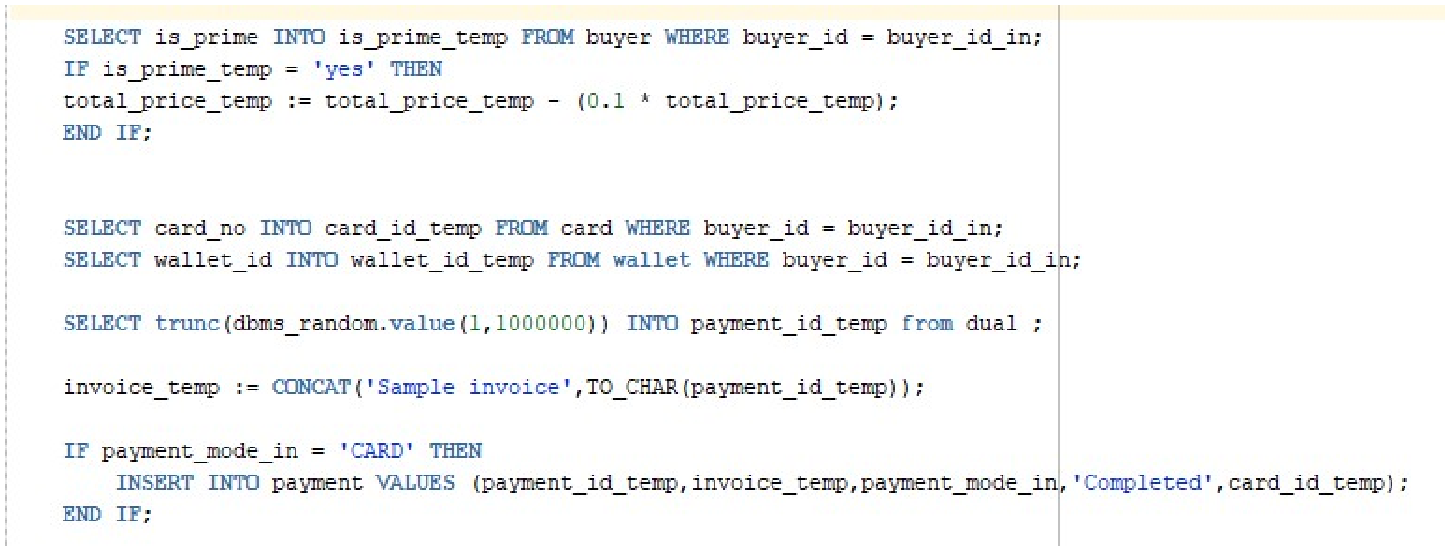
```
CREATE OR REPLACE PROCEDURE set_default_contact_detail (
address_id_var IN INTEGER,
buyer_id_var IN VARCHAR
) AS
BEGIN
  update contact_detail set is_default = 1 where email = buyer_id_var
    and address_id = address_id_var;
END set_default_contact_detail;

CREATE OR REPLACE PROCEDURE update_prime_membership (
buyer_id_var IN VARCHAR
) AS
BEGIN
  update buyer set is_prime = 'Yes', prime_expiry_date = add_months(sysdate , 12)
  WHERE buyer_id = buyer_id_var;
END update_prime_membership;

CREATE OR REPLACE PROCEDURE cancel_prime_membership (
buyer_id_var IN VARCHAR
) AS
BEGIN
  update buyer set is_prime = 'No', prime_expiry_date = null where
  buyer_id = buyer_id_var;
END cancel_prime_membership;
```



```
CREATE OR REPLACE PROCEDURE contact_customer_support
(
  buyer_id_var IN VARCHAR
) AS
   cust_rep_id  NUMBER;
   BEGIN
       select customer_rep_id into cust_rep_id from customer_support where assigned_to is null fetch first 1 rows only;
       update amazon_user set customer_rep_id = cust_rep_id where email = buyer_id_var;
       update customer_support set assigned_to = buyer_id_var where customer_rep_id = cust_rep_id;
END contact_customer_support;
```



```
CREATE OR REPLACE PROCEDURE place_order (
    order_id IN INTEGER,
    buyer_id_in IN VARCHAR,
    payment_mode_in IN VARCHAR
) AS
    card_id_temp INTEGER;
    wallet_id_temp INTEGER;
    address_id_temp INTEGER;
    total_price_temp NUMBER := 0;
    curr_price_temp NUMBER;
    total_qty_temp NUMBER := 0;
    available_units_temp NUMBER;
    is_prime_temp VARCHAR(255);
    payment_id_temp INTEGER;
    invoice_temp VARCHAR(255);

    CURSOR products_cursor IS
        SELECT product_id FROM product_shopping_cart WHERE buyer_id = buyer_id_in;

        product_id_temp INTEGER;
        BEGIN
            OPEN products_cursor;
            LOOP
                FETCH products_cursor INTO product_id_temp;
                EXIT WHEN products_cursor%notfound;

                SELECT price, available_units INTO curr_price_temp, available_units_temp FROM product
                WHERE product_id = product_id_temp;

                IF available_units_temp > 0 THEN
                    total_price_temp := ( total_price_temp + curr_price_temp );
                    total_qty_temp := total_qty_temp + 1;
                END IF;
            END LOOP;
    CLOSE products_cursor;
```

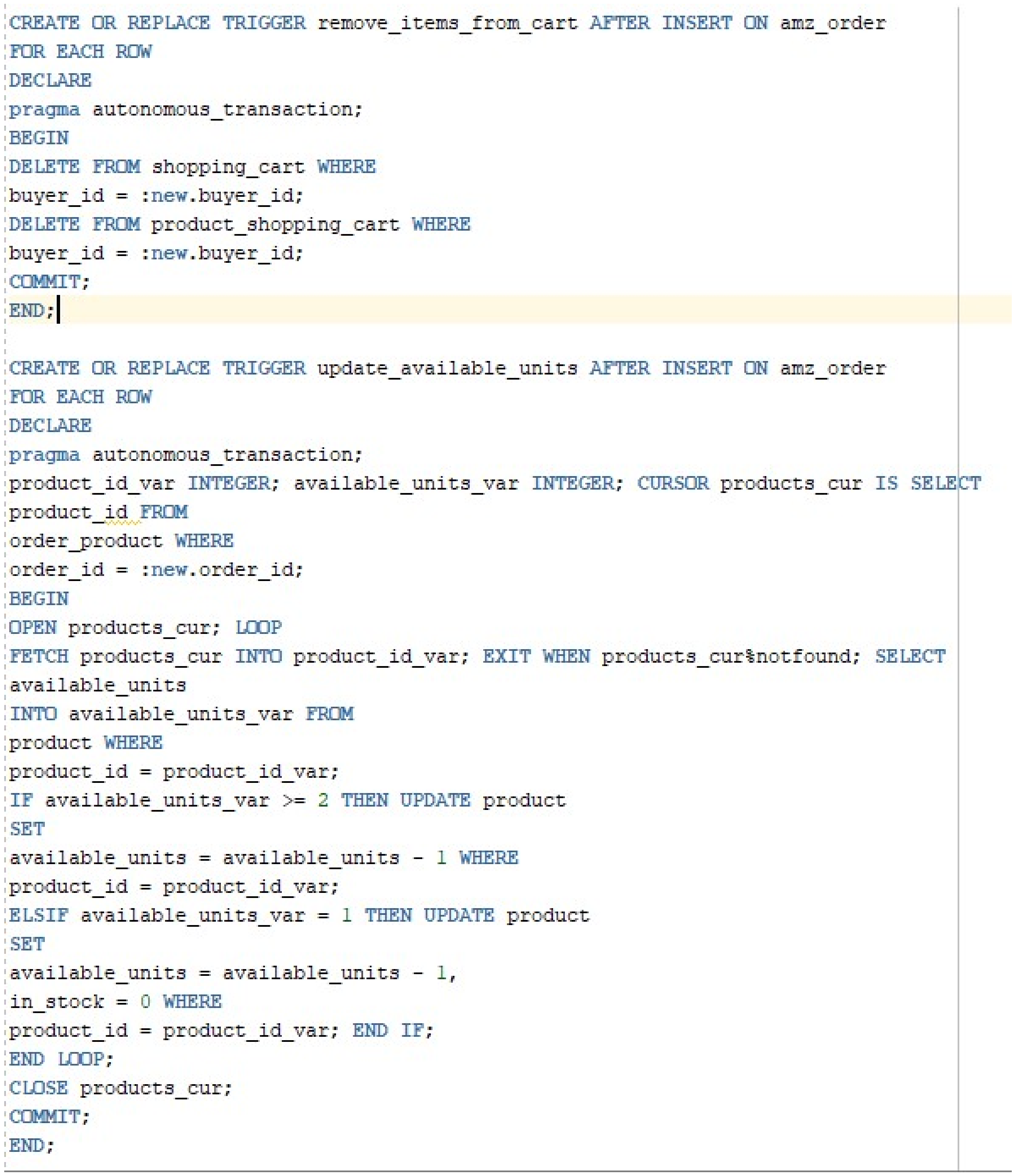


```
    SELECT is_prime INTO is_prime_temp FROM buyer WHERE buyer_id = buyer_id_in;
    IF is_prime_temp = 'yes' THEN
    total_price_temp := total_price_temp - (0.1 * total_price_temp);
    END IF;


    SELECT card_no INTO card_id_temp FROM card WHERE buyer_id = buyer_id_in;
    SELECT wallet_id INTO wallet_id_temp FROM wallet WHERE buyer_id = buyer_id_in;

    SELECT trunc(dbms_random.value(1,1000000)) INTO payment_id_temp from dual ;

    invoice_temp := CONCAT('Sample invoice',TO_CHAR(payment_id_temp));

    IF payment_mode_in = 'CARD' THEN
        INSERT INTO payment VALUES (payment_id_temp,invoice_temp,payment_mode_in,'Completed',card_id_temp);
    END IF;
    IF payment_mode_in = 'WALLET' THEN
        INSERT INTO payment VALUES (payment_id_temp,invoice_temp,payment_mode_in,'Completed',wallet_id_temp);
    END IF;

     SELECT address_id INTO address_id_temp FROM contact_detail WHERE email = buyer_id_in and is_default=1;

INSERT INTO amz_order VALUES (
order_id,
buyer_id_in,
payment_id_temp,
total_price_temp,
sysdate,
address_id_temp,
add_months(DATE '2019-11-28', 1),
'Completed',
total_qty_temp
);
END place_order;
```

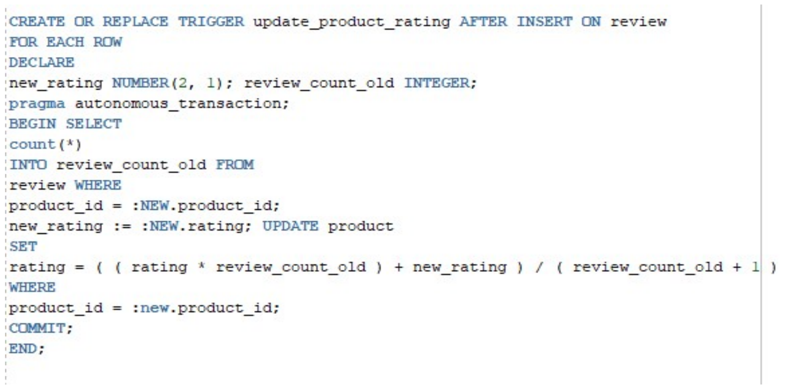
```
CREATE OR REPLACE PROCEDURE cancel_order (
order_id_in IN VARCHAR
) AS
total_price_temp INTEGER;
payment_mode_temp VARCHAR(255);
buyer_id_temp VARCHAR(255);
BEGIN
  update amz_order set status='Cancelled' where order_id = order_id_in;

  select total_price, buyer_id into total_price_temp, buyer_id_temp from amz_order where order_id = order_id_in;

  update wallet set balance = balance + total_price_temp where buyer_id = buyer_id_temp;

END cancel_order;
```

```
CREATE OR REPLACE PROCEDURE add_coupon_to_wallet
(
  buyer_id_var IN VARCHAR,
  coupon_id_var IN NUMBER
) AS
   coupon_value_var INTEGER;
   wallet_id_var  NUMBER;
   BEGIN
       select wallet_id into wallet_id_var from wallet where buyer_id = buyer_id_var;
       select coupon_value into coupon_value_var from coupon where coupon_id = coupon_id_var;
       update wallet set balance = balance + coupon_value_var where wallet_id = wallet_id_var;
       update coupon set buyer_id = buyer_id_var , wallet_id = wallet_id_var where coupon_id = coupon_id_var;
END add_coupon_to_wallet;
```

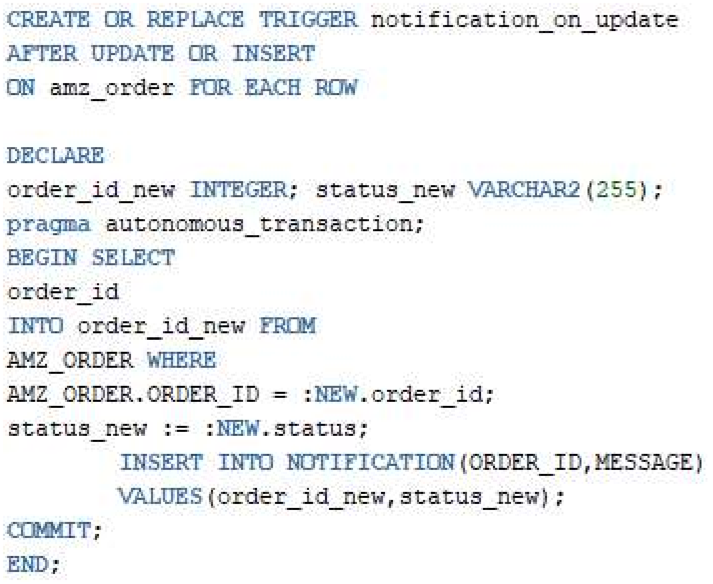
Triggers: -



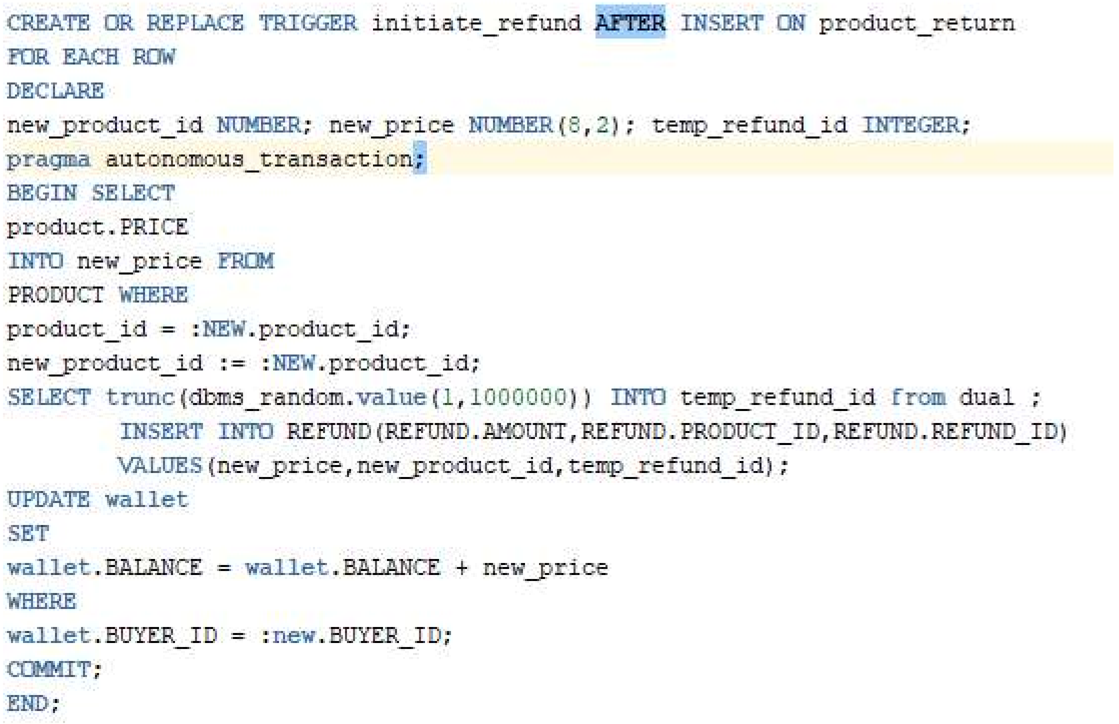
```
CREATE OR REPLACE TRIGGER remove_items_from_cart AFTER INSERT ON amz_order
FOR EACH ROW
DECLARE
pragma autonomous_transaction;
BEGIN
DELETE FROM shopping_cart WHERE
buyer_id = :new.buyer_id;
DELETE FROM product_shopping_cart WHERE
buyer_id = :new.buyer_id;
COMMIT;
END;

CREATE OR REPLACE TRIGGER update_available_units AFTER INSERT ON amz_order
FOR EACH ROW
DECLARE
pragma autonomous_transaction;
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;
COMMIT;
END;
```



```
CREATE OR REPLACE TRIGGER update_product_rating AFTER INSERT ON review
FOR EACH ROW
DECLARE
new_rating NUMBER(2, 1); review_count_old INTEGER;
pragma autonomous_transaction;
BEGIN SELECT
count(*)
INTO review_count_old FROM
review WHERE
product_id = :NEW.product_id;
new_rating := :NEW.rating; UPDATE product
SET
rating = ( ( rating * review_count_old ) + new_rating ) / ( review_count_old + 1 )
WHERE
product_id = :new.product_id;
COMMIT;
END;
```

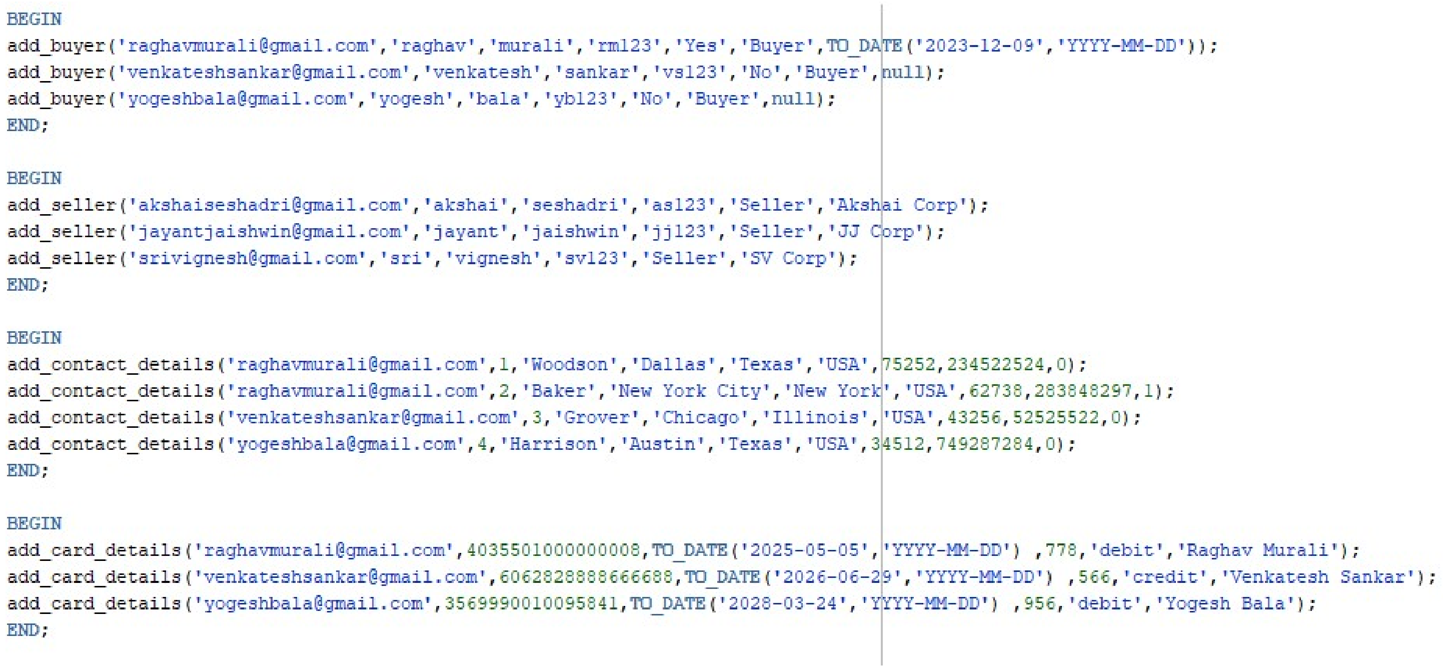


```
CREATE OR REPLACE TRIGGER notification_on_update
AFTER UPDATE OR INSERT
ON amz_order FOR EACH ROW

DECLARE
order_id_new INTEGER; status_new VARCHAR2(255);
pragma autonomous_transaction;
BEGIN SELECT
order_id
INTO order_id_new FROM
AMZ_ORDER WHERE
AMZ_ORDER.ORDER_ID = :NEW.order_id;
status_new := :NEW.status;
        INSERT INTO NOTIFICATION(ORDER_ID,MESSAGE)
        VALUES(order_id_new,status_new);
COMMIT;
END;
```

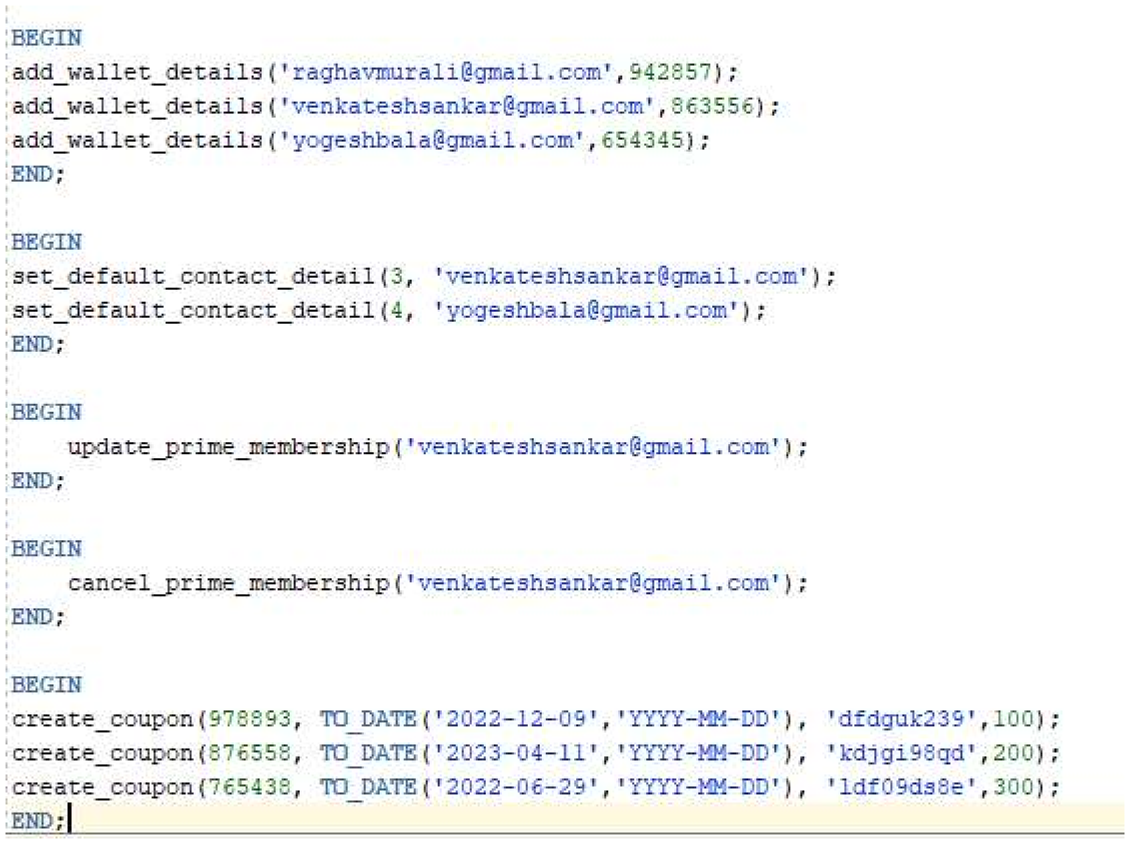


```
CREATE OR REPLACE TRIGGER initiate_refund AFTER INSERT ON product_return
FOR EACH ROW
DECLARE
new_product_id NUMBER; new_price NUMBER(8,2); temp_refund_id INTEGER;
pragma autonomous_transaction;
BEGIN SELECT
product.PRICE
INTO new_price FROM
PRODUCT WHERE
product_id = :NEW.product_id;
new_product_id := :NEW.product_id;
SELECT trunc(dbms_random.value(1,1000000)) INTO temp_refund_id from dual ;
        INSERT INTO REFUND(REFUND.AMOUNT,REFUND.PRODUCT_ID,REFUND.REFUND_ID)
        VALUES(new_price,new_product_id,temp_refund_id);
UPDATE wallet
SET
wallet.BALANCE = wallet.BALANCE + new_price
WHERE
wallet.BUYER_ID = :new.BUYER_ID;
COMMIT;
END;
```

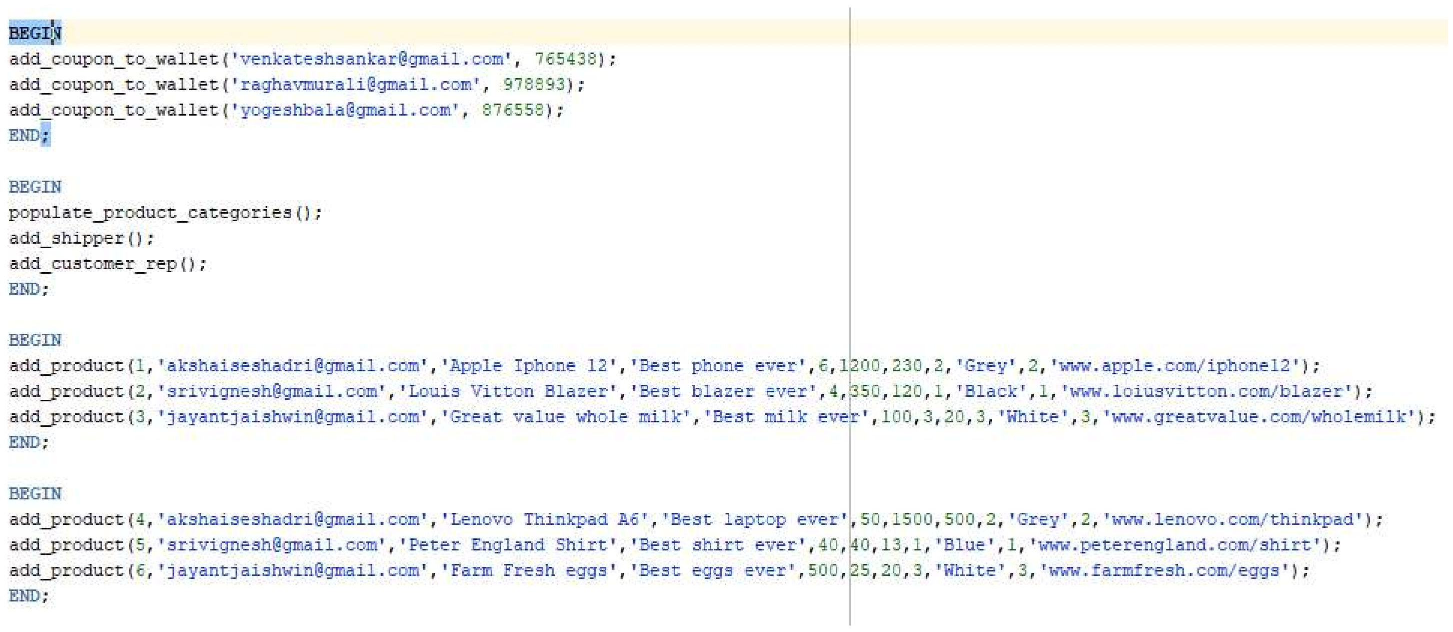
Calling the procedures: -



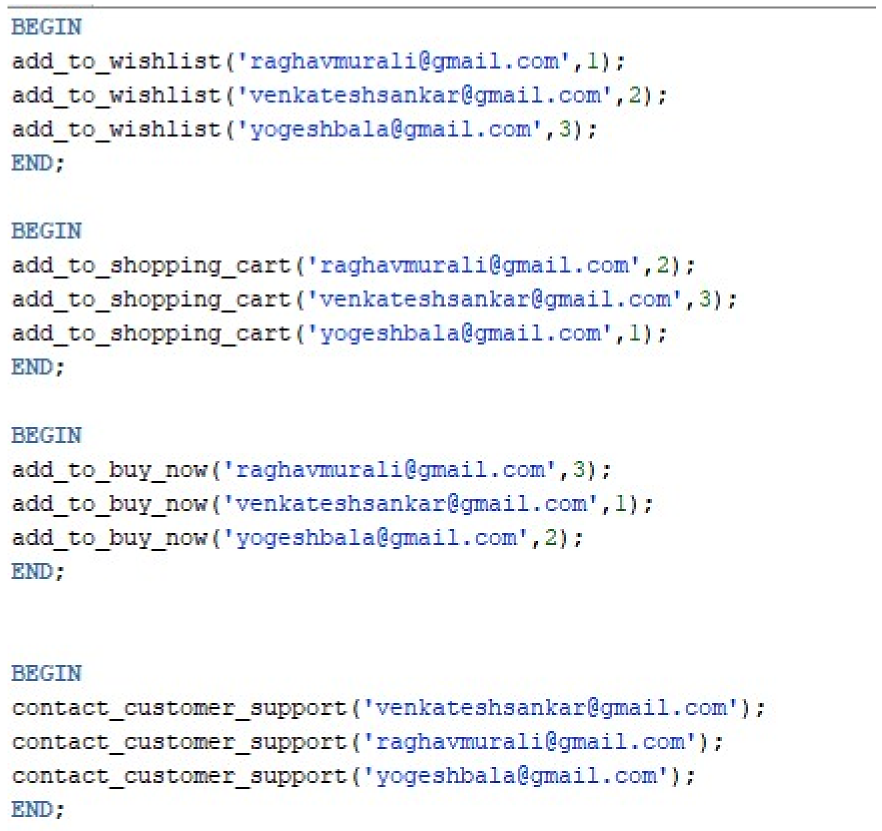
```
BEGIN
add_buyer('raghavmurali@gmail.com','raghav','murali','rm123','Yes','Buyer',TO_DATE('2023-12-09','YYYY-MM-DD'));
add_buyer('venkateshsankar@gmail.com','venkatesh','sankar','vs123','No','Buyer',null);
add_buyer('yogeshbala@gmail.com','yogesh','bala','yb123','No','Buyer',null);
END;

BEGIN
add_seller('akshaiseshadri@gmail.com','akshai','seshadri','as123','Seller','Akshai Corp');
add_seller('jayantjaishwin@gmail.com','jayant','jaishwin','jj123','Seller','JJ Corp');
add_seller('srivignesh@gmail.com','sri','vignesh','sv123','Seller','SV Corp');
END;

BEGIN
add_contact_details('raghavmurali@gmail.com',1,'Woodson','Dallas','Texas','USA',75252,234522524,0);
add_contact_details('raghavmurali@gmail.com',2,'Baker','New York City','New York','USA',62738,283848297,1);
add_contact_details('venkateshsankar@gmail.com',3,'Grover','Chicago','Illinois','USA',43256,52525522,0);
add_contact_details('yogeshbala@gmail.com',4,'Harrison','Austin','Texas','USA',34512,749287284,0);
END;

BEGIN
add_card_details('raghavmurali@gmail.com',4035501000000008,TO_DATE('2025-05-05','YYYY-MM-DD') ,778,'debit','Raghav Murali');
add_card_details('venkateshsankar@gmail.com',6062828888666688,TO_DATE('2026-06-29','YYYY-MM-DD') ,566,'credit','Venkatesh Sankar');
add_card_details('yogeshbala@gmail.com',3569990010095841,TO_DATE('2028-03-24','YYYY-MM-DD') ,956,'debit','Yogesh Bala');
END;
```



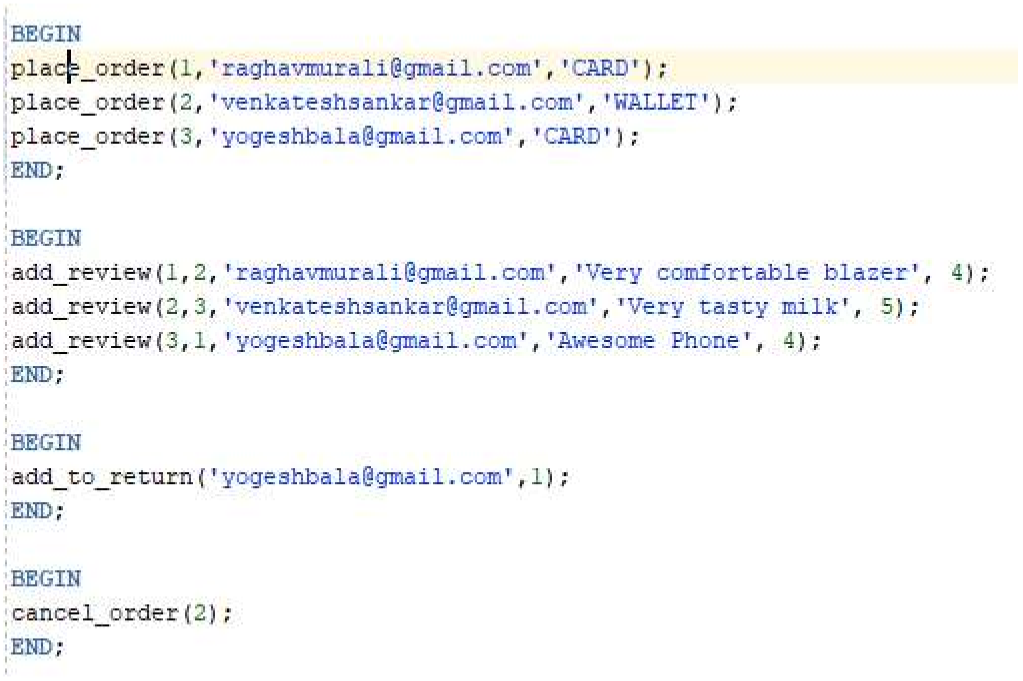
```
BEGIN
add_wallet_details('raghavmurali@gmail.com',942857);
add_wallet_details('venkateshsankar@gmail.com',863556);
add_wallet_details('yogeshbala@gmail.com',654345);
END;

BEGIN
set_default_contact_detail(3, 'venkateshsankar@gmail.com');
set_default_contact_detail(4, 'yogeshbala@gmail.com');
END;

BEGIN
    update_prime_membership('venkateshsankar@gmail.com');
END;

BEGIN
    cancel_prime_membership('venkateshsankar@gmail.com');
END;

BEGIN
create_coupon(978893, TO_DATE('2022-12-09','YYYY-MM-DD'), 'dfdguk239',100);
create_coupon(876558, TO_DATE('2023-04-11','YYYY-MM-DD'), 'kdjgi98qd',200);
create_coupon(765438, TO_DATE('2022-06-29','YYYY-MM-DD'), 'ldf09ds8e',300);
END;
```



```
BEGIN
add_coupon_to_wallet('venkateshsankar@gmail.com', 765438);
add_coupon_to_wallet('raghavmurali@gmail.com', 978893);
add_coupon_to_wallet('yogeshbala@gmail.com', 876558);
END;

BEGIN
populate_product_categories();
add_shipper();
add_customer_rep();
END;

BEGIN
add_product(1,'akshaiseshadri@gmail.com','Apple Iphone 12','Best phone ever',6,1200,230,2,'Grey',2,'www.apple.com/iphone12');
add_product(2,'srivignesh@gmail.com','Louis Vitton Blazer','Best blazer ever',4,350,120,1,'Black',1,'www.loiusvitton.com/blazer');
add_product(3,'jayantjaishwin@gmail.com','Great value whole milk','Best milk ever',100,3,20,3,'White',3,'www.greatvalue.com/wholemilk');
END;

BEGIN
add_product(4,'akshaiseshadri@gmail.com','Lenovo Thinkpad A6','Best laptop ever',50,1500,500,2,'Grey',2,'www.lenovo.com/thinkpad');
add_product(5,'srivignesh@gmail.com','Peter England Shirt','Best shirt ever',40,40,13,1,'Blue',1,'www.peterengland.com/shirt');
add_product(6,'jayantjaishwin@gmail.com','Farm Fresh eggs','Best eggs ever',500,25,20,3,'White',3,'www.farmfresh.com/eggs');
END;
```



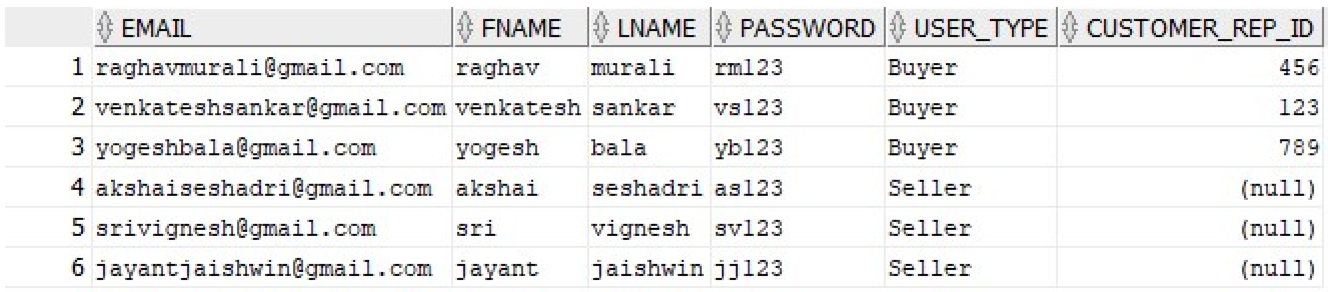
```
BEGIN
add_to_wishlist('raghavmurali@gmail.com',1);
add_to_wishlist('venkateshsankar@gmail.com',2);
add_to_wishlist('yogeshbala@gmail.com',3);
END;

BEGIN
add_to_shopping_cart('raghavmurali@gmail.com',2);
add_to_shopping_cart('venkateshsankar@gmail.com',3);
add_to_shopping_cart('yogeshbala@gmail.com',1);
END;

BEGIN
add_to_buy_now('raghavmurali@gmail.com',3);
add_to_buy_now('venkateshsankar@gmail.com',1);
add_to_buy_now('yogeshbala@gmail.com',2);
END;


BEGIN
contact_customer_support('venkateshsankar@gmail.com');
contact_customer_support('raghavmurali@gmail.com');
contact_customer_support('yogeshbala@gmail.com');
END;
```



```
BEGIN
place_order(1,'raghavmurali@gmail.com','CARD');
place_order(2,'venkateshsankar@gmail.com','WALLET');
place_order(3,'yogeshbala@gmail.com','CARD');
END;

BEGIN
add_review(1,2,'raghavmurali@gmail.com','Very comfortable blazer', 4);
add_review(2,3,'venkateshsankar@gmail.com','Very tasty milk', 5);
add_review(3,1,'yogeshbala@gmail.com','Awesome Phone', 4);
END;

BEGIN
add_to_return('yogeshbala@gmail.com',1);
END;

BEGIN
cancel_order(2);
END;
```

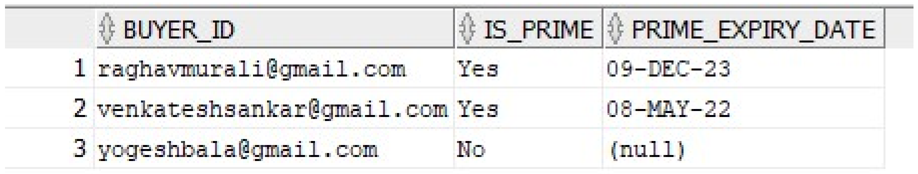
Table data after implementation: -

Amazon_user:-



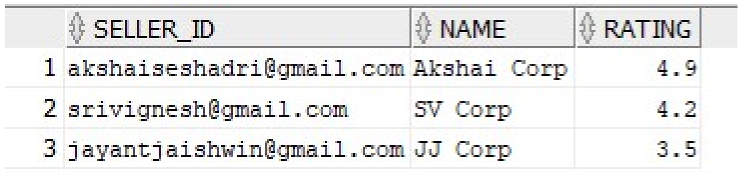
	EMAIL	FNAME	LNAME	PASSWORD	USER_TYPE	CUSTOMER_REP_ID
1	raghavmurali@gmail.com	raghav	murali	rm123	Buyer	456
2	venkateshsankar@gmail.com	venkatesh	sankar	vs123	Buyer	123
3	yogeshbala@gmail.com	yogesh	bala	yb123	Buyer	789
4	akshaiseshadri@gmail.com	akshai	seshadri	as123	Seller	(null)
5	srivignesh@gmail.com	sri	vignesh	sv123	Seller	(null)
6	jayantjaishwin@gmail.com	jayant	jaishwin	jj123	Seller	(null)

Buyer: -



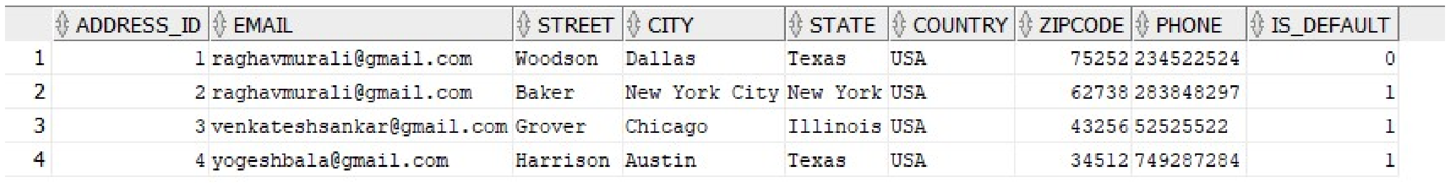
	BUYER_ID	IS_PRIME	PRIME_EXPIRY_DATE
1	raghavmurali@gmail.com	Yes	09-DEC-23
2	venkateshsankar@gmail.com	Yes	08-MAY-22
3	yogeshbala@gmail.com	No	(null)

Seller: -



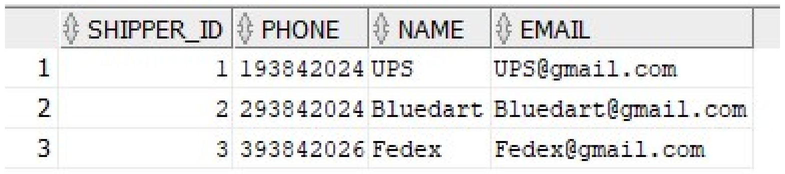
	SELLER_ID	NAME	RATING
1	akshaiseshadri@gmail.com	Akshai Corp	4.9
2	srivignesh@gmail.com	SV Corp	4.2
3	jayantjaishwin@gmail.com	JJ Corp	3.5

Contact_detail: -



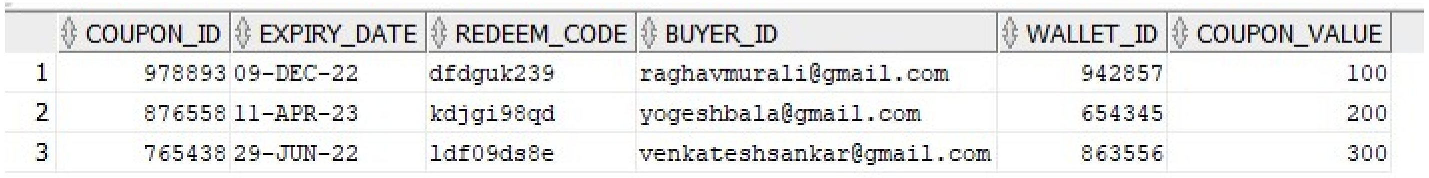
	ADDRESS_ID	EMAIL	STREET	CITY	STATE	COUNTRY	ZIPCODE	PHONE	IS_DEFAULT
1	1	raghavmurali@gmail.com	Woodson	Dallas	Texas	USA	75252	234522524	0
2	2	raghavmurali@gmail.com	Baker	New York City	New York	USA	62738	283848297	1
3	3	venkateshsankar@gmail.com	Grover	Chicago	Illinois	USA	43256	52525522	1
4	4	yogeshbala@gmail.com	Harrison	Austin	Texas	USA	34512	749287284	1

Shipper: -



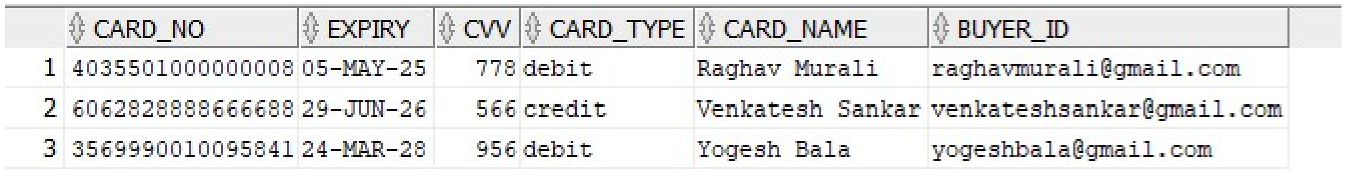
	SHIPPER_ID	PHONE	NAME	EMAIL
1	1	193842024	UPS	UPS@gmail.com
2	2	293842024	Bluedart	Bluedart@gmail.com
3	3	393842026	Fedex	Fedex@gmail.com

Coupon: -



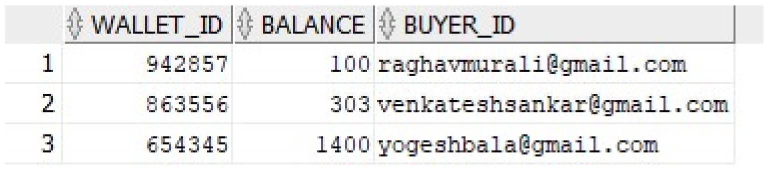
	COUPON_ID	EXPIRY_DATE	REDEEM_CODE	BUYER_ID	WALLET_ID	COUPON_VALUE
1	978893	09-DEC-22	dfdguk239	raghavmurali@gmail.com	942857	100
2	876558	11-APR-23	kdjgi98qd	yogeshbala@gmail.com	654345	200
3	765438	29-JUN-22	ldf09ds8e	venkateshsankar@gmail.com	863556	300

Card: -



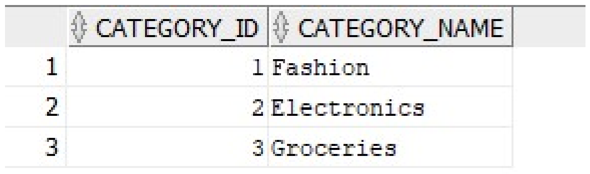
	CARD_NO	EXPIRY	CVV	CARD_TYPE	CARD_NAME	BUYER_ID
1	4035501000000008	05-MAY-25	778	debit	Raghav Murali	raghavmurali@gmail.com
2	6062828888666688	29-JUN-26	566	credit	Venkatesh Sankar	venkateshsankar@gmail.com
3	3569990010095841	24-MAR-28	956	debit	Yogesh Bala	yogeshbala@gmail.com

Wallet: -



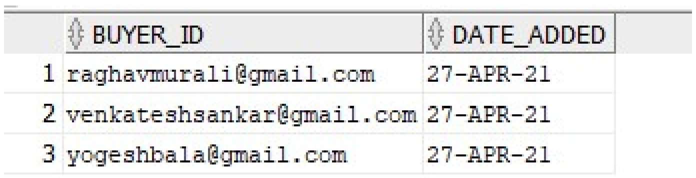
	WALLET_ID	BALANCE	BUYER_ID
1	942857	100	raghavmurali@gmail.com
2	863556	303	venkateshsankar@gmail.com
3	654345	1400	yogeshbala@gmail.com

Category: -



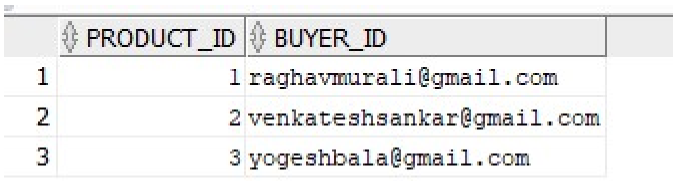
	CATEGORY_ID	CATEGORY_NAME
1	1	Fashion
2	2	Electronics
3	3	Groceries

Wishlist: -



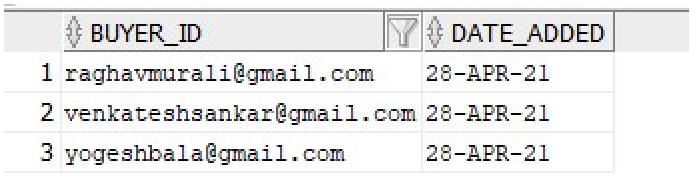
	BUYER_ID	DATE_ADDED
1	raghavmurali@gmail.com	27-APR-21
2	venkateshsankar@gmail.com	27-APR-21
3	yogeshbala@gmail.com	27-APR-21

Product_wishlist: -



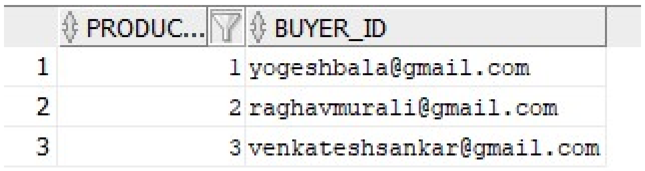
	PRODUCT_ID	BUYER_ID
1	1	raghavmurali@gmail.com
2	2	venkateshsankar@gmail.com
3	3	yogeshbala@gmail.com

Shopping_cart: -



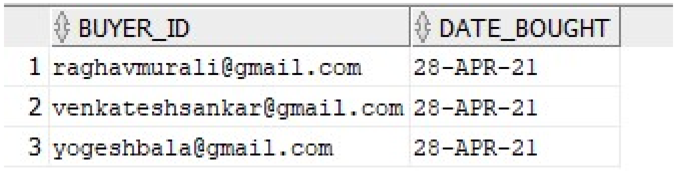
	BUYER_ID	DATE_ADDED
1	raghavmurali@gmail.com	28-APR-21
2	venkateshsankar@gmail.com	28-APR-21
3	yogeshbala@gmail.com	28-APR-21

Product_shopping_cart: -



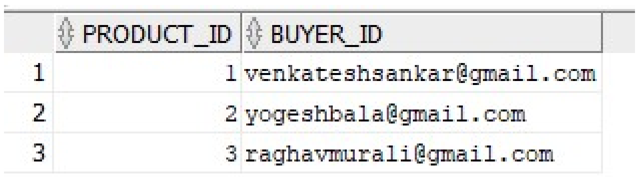
	PRODUC...	BUYER_ID
1	1	yogeshbala@gmail.com
2	2	raghavmurali@gmail.com
3	3	venkateshsankar@gmail.com

Buy_now :-



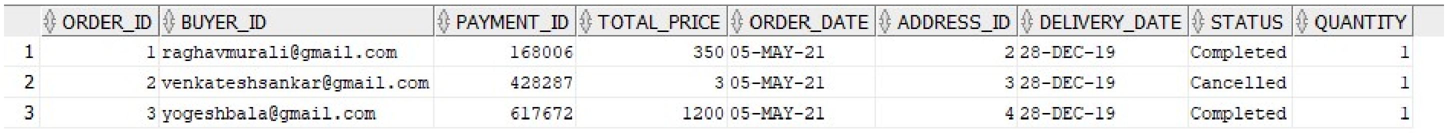
	BUYER_ID	DATE_BOUGHT
1	raghavmurali@gmail.com	28-APR-21
2	venkateshsankar@gmail.com	28-APR-21
3	yogeshbala@gmail.com	28-APR-21

Product_buy_now: -



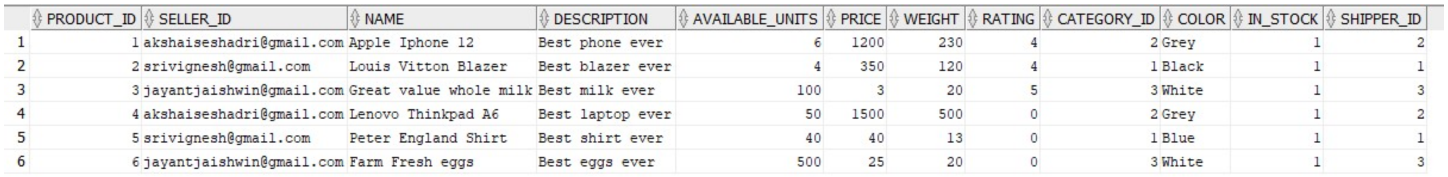
	PRODUCT_ID	BUYER_ID
1	1	venkateshsankar@gmail.com
2	2	yogeshbala@gmail.com
3	3	raghavmurali@gmail.com

Amz_order: -



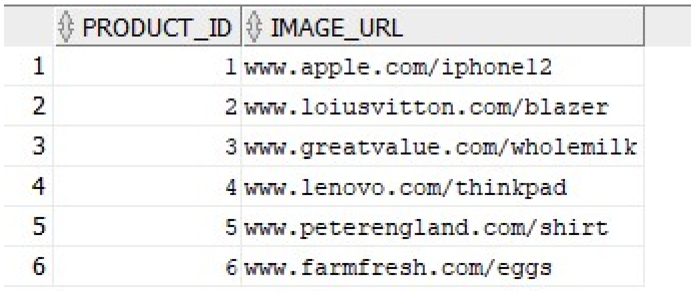
	ORDER_ID	BUYER_ID	PAYMENT_ID	TOTAL_PRICE	ORDER_DATE	ADDRESS_ID	DELIVERY_DATE	STATUS	QUANTITY
1	1	raghavmurali@gmail.com	168006	350	05-MAY-21	2	28-DEC-19	Completed	1
2	2	venkateshsankar@gmail.com	428287	3	05-MAY-21	3	28-DEC-19	Cancelled	1
3	3	yogeshbala@gmail.com	617672	1200	05-MAY-21	4	28-DEC-19	Completed	1

Product: -



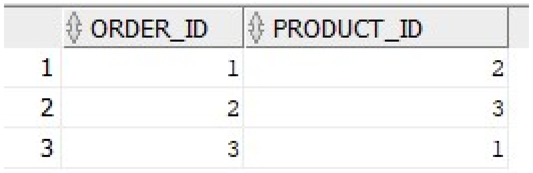
	PRODUCT_ID	SELLER_ID	NAME	DESCRIPTION	AVAILABLE_UNITS	PRICE	WEIGHT	RATING	CATEGORY_ID	COLOR	IN_STOCK	SHIPPER_ID
1	1	akshaiseshadri@gmail.com	Apple Iphone 12	Best phone ever	6	1200	230	4	2	Grey	1	2
2	2	srivignesh@gmail.com	Louis Vitton Blazer	Best blazer ever	4	350	120	4	1	Black	1	1
3	3	jayantjaishwin@gmail.com	Great value whole milk	Best milk ever	100	3	20	5	3	White	1	3
4	4	akshaiseshadri@gmail.com	Lenovo Thinkpad A6	Best laptop ever	50	1500	500	0	2	Grey	1	2
5	5	srivignesh@gmail.com	Peter England Shirt	Best shirt ever	40	40	13	0	1	Blue	1	1
6	6	jayantjaishwin@gmail.com	Farm Fresh eggs	Best eggs ever	500	25	20	0	3	White	1	3

Product_image: -



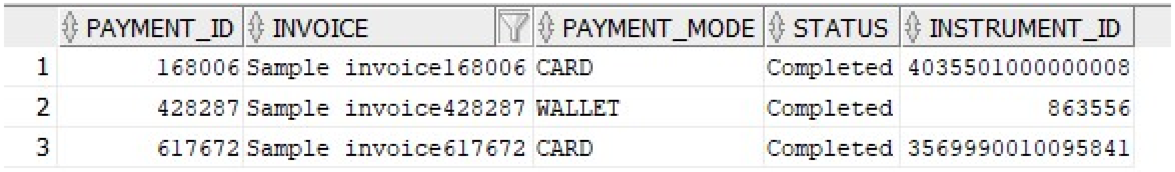
	PRODUCT_ID	IMAGE_URL
1	1	www.apple.com/iphone12
2	2	www.loiusvitton.com/blazer
3	3	www.greatvalue.com/wholemilk
4	4	www.lenovo.com/thinkpad
5	5	www.peterengland.com/shirt
6	6	www.farmfresh.com/eggs

Order_product: -



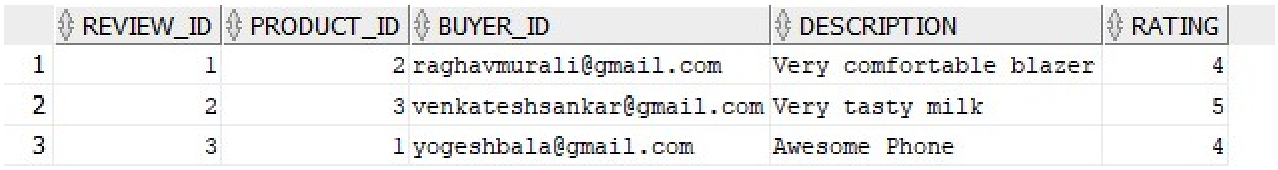
	ORDER_ID	PRODUCT_ID
1	1	2
2	2	3
3	3	1

Payment: -



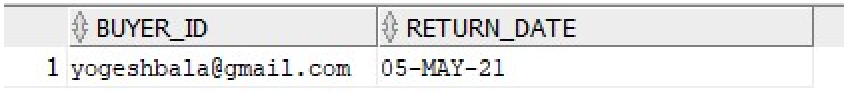
	PAYMENT_ID	INVOICE	PAYMENT_MODE	STATUS	INSTRUMENT_ID
1	168006	Sample invoice168006	CARD	Completed	4035501000000008
2	428287	Sample invoice428287	WALLET	Completed	863556
3	617672	Sample invoice617672	CARD	Completed	3569990010095841

Review: -



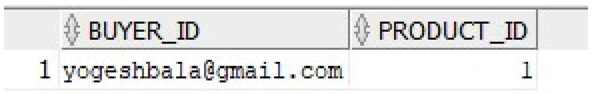
	REVIEW_ID	PRODUCT_ID	BUYER_ID	DESCRIPTION	RATING
1	1	2	raghavmurali@gmail.com	Very comfortable blazer	4
2	2	3	venkateshsankar@gmail.com	Very tasty milk	5
3	3	1	yogeshbala@gmail.com	Awesome Phone	4

Return: -



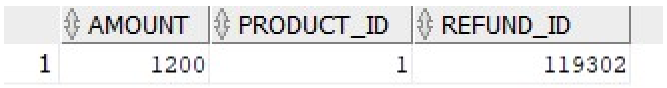
	BUYER_ID	RETURN_DATE
1	yogeshbala@gmail.com	05-MAY-21

Product_return: -



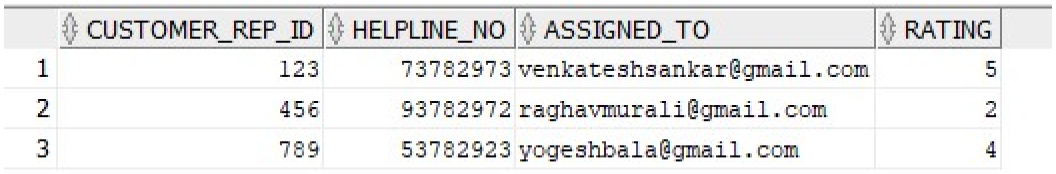
	BUYER_ID	PRODUCT_ID
1	yogeshbala@gmail.com	1

Refund: -



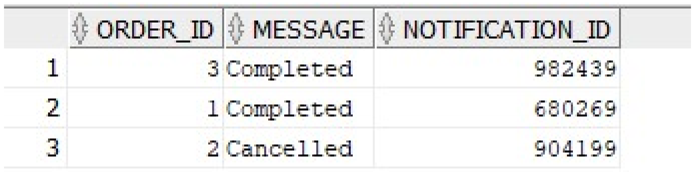
	AMOUNT	PRODUCT_ID	REFUND_ID
1	1200	1	119302

Customer_support: -



	CUSTOMER_REP_ID	HELPLINE_NO	ASSIGNED_TO	RATING
1	123	73782973	venkateshsankar@gmail.com	5
2	456	93782972	raghavmurali@gmail.com	2
3	789	53782923	yogeshbala@gmail.com	4

Notification: -



	ORDER_ID	MESSAGE	NOTIFICATION_ID
1	3	Completed	982439
2	1	Completed	680269
3	2	Cancelled	904199