PROJECT REPORT

AMAZON DATABASE DESIGN

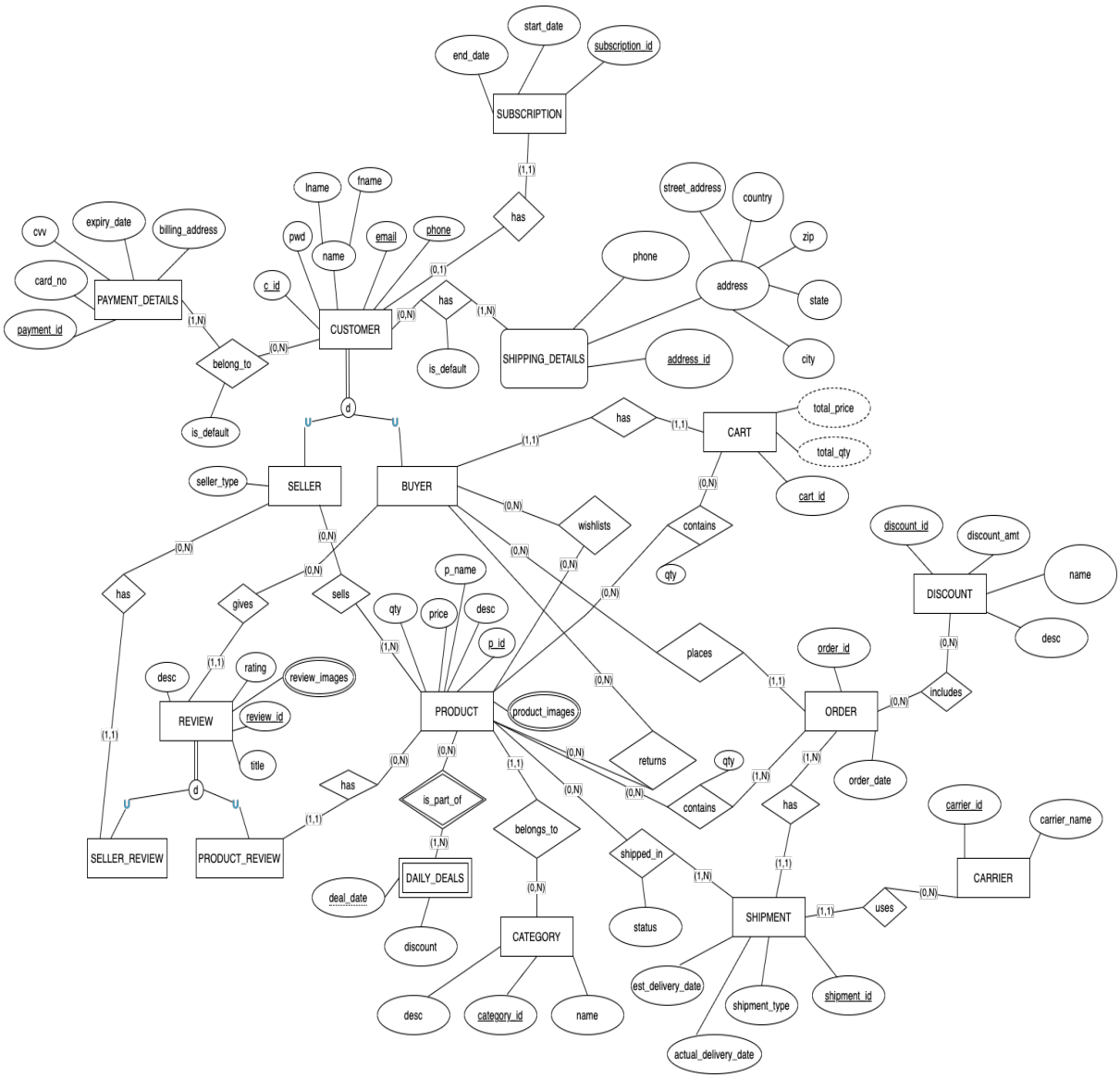
TEAM MEMBERS

MARY MATHEWS	**MXM190127**
KRISHNA VAMSI RIMMALAPUDI	**KXR190033**
YASASWI DEVI TIYYAGURA	**YXT200010**

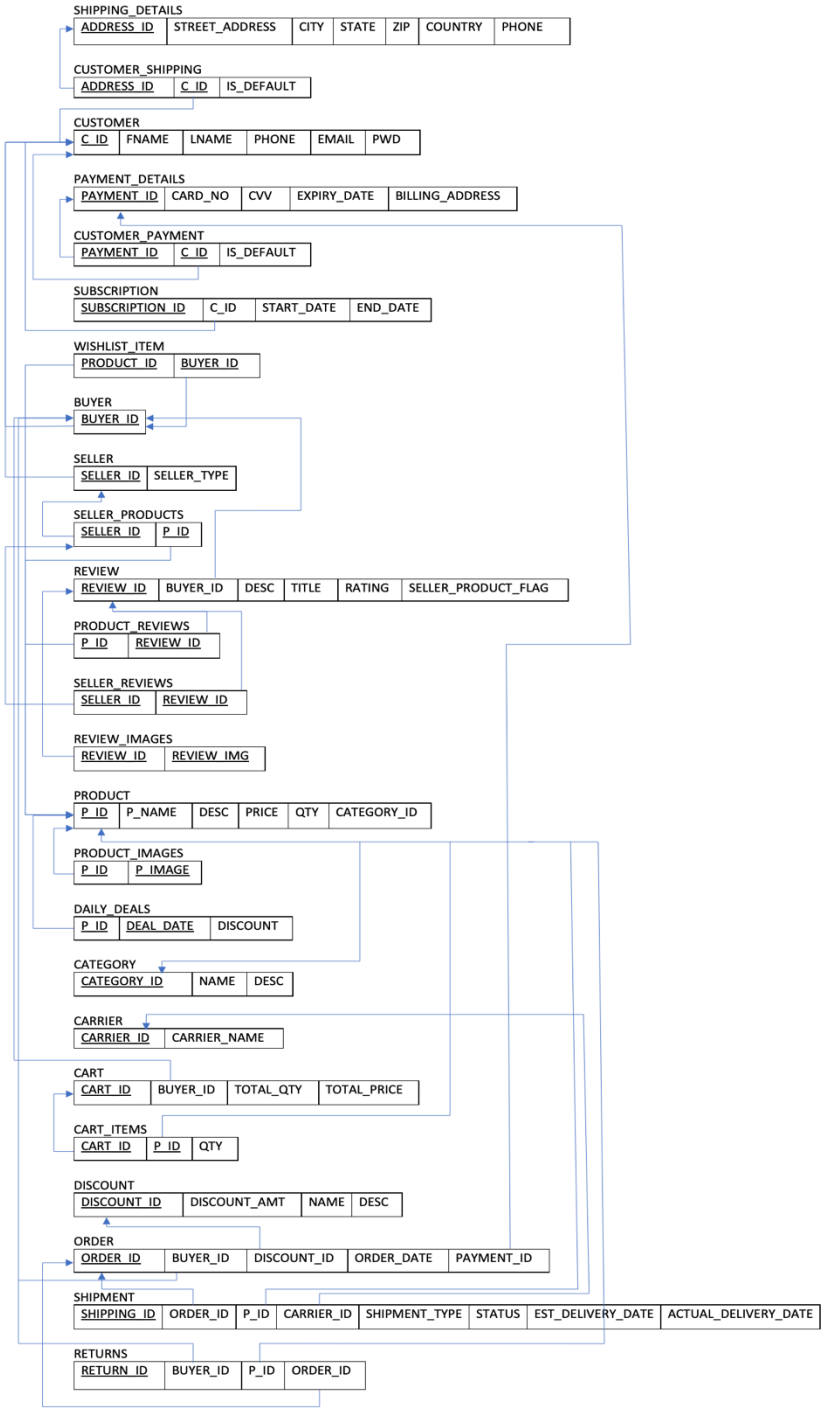
Functional Requirements:

1. A user can sign-up either as a buyer or a seller.
2. A user can log in to their account.
3. A user can save zero or more shipping addresses.
4. An order can be placed only if the user has added a shipping address.
5. A user can save one or more contact numbers.
6. A user can save zero or more payment methods.
7. An order can be placed only if the user has added a payment method.
8. There is a pre-defined set of categories.
9. A seller can add products.
10. A seller may be a brand owner or a reseller.
11. Each product belongs to exactly one category.
12. The product details can include:
    a. Required –
        - Product name
        - One or more images
        - Product description
        - Price
        - Quantity in stock
    b. Optional –
        - Product rating
        - Product reviews
        - Product discount
13. A buyer can add products to multiple wish-lists.
14. A buyer can add products to a cart.
15. Each buyer has exactly one cart.
16. A buyer can place an order with the products in their cart.
17. A buyer can review a product.
18. A buyer may only give one review per product.
19. A buyer can only review products they have purchased.
20. A buyer can review a seller.
21. A buyer may only give one review per seller.
22. A buyer can only review sellers from which they have purchased products.
23. A buyer can return products.
24. A user can have a prime subscription.
25. Each prime subscription has a start date and an expiry date.
26. A product can be a part of a daily deals discount.
27. An order is shipped in one or more shipments.
28. Products within an order may be shipped in different shipments.
29. Shipments within an order may use different carriers.

EER Diagram:


start_date
end_date
subscription_id
SUBSCRIPTION
(1,1)
has
lname
fname
pwd
name
email
phone
c_id
CUSTOMER
(0,N)
has
(1,N)
(0,1)
is_default
cvv
expiry_date
billing_address
card_no
PAYMENT_DETAILS
payment_id
(1,N)
belong_to
(0,N)
is_default
d
U
U
SELLER
seller_type
BUYER
has
(1,1)
CART
total_price
total_qty
cart_id
(0,N)
contains
qty
wishlists
(0,N)
(0,N)
(0,N)
(0,N)
(0,N)
has
gives
sells
(0,N)
(0,N)
(0,N)
(1,N)
qty
price
desc
p_name
p_id
(1,1)
(1,1)
desc
rating
review_images
review_id
title
REVIEW
d
U
U
SELLER_REVIEW
PRODUCT_REVIEW
(1,1)
PRODUCT
product_images
(0,N)
has
(0,N)
is_part_of
(1,N)
DAILY_DEALS
deal_date
discount
(1,1)
belongs_to
(0,N)
CATEGORY
desc
category_id
name
(0,N)
returns
(0,N)
(0,N)
qty
contains
(1,N)
shipped_in
(1,N)
status
places
(1,1)
order_id
ORDER
(0,N)
includes
(0,N)
order_date
(1,N)
has
(1,1)
SHIPMENT
est_delivery_date
shipment_type
shipment_id
actual_delivery_date
(1,1)
uses
(0,N)
carrier_id
carrier_name
CARRIER
discount_id
discount_amt
name
DISCOUNT
desc

Schema Diagram:



Database Creation Statements:

```
CREATE TABLE CUSTOMER (
    C_ID    CHAR(15),
    FNAME   VARCHAR(30) NOT NULL,
    LNAME   VARCHAR(30) NOT NULL,
    PHONE   CHAR(10) UNIQUE NOT NULL,
    EMAIL   VARCHAR(60) UNIQUE NOT NULL,
    PWD     VARCHAR(60) NOT NULL,
    PRIMARY KEY ( C_ID )
);

CREATE TABLE SHIPPING_DETAILS (
    ADDRESS_ID      NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    STREET_ADDRESS  VARCHAR(100) NOT NULL,
    CITY            VARCHAR(100) NOT NULL,
    STATE           VARCHAR(100) NOT NULL,
    ZIP             VARCHAR(10) NOT NULL,
    COUNTRY         VARCHAR(60) NOT NULL,
    PHONE           CHAR(10) NOT NULL,
    PRIMARY KEY ( ADDRESS_ID )
);

CREATE TABLE CUSTOMER_SHIPPING (
    ADDRESS_ID  NUMBER,
    C_ID        CHAR(15),
    IS_DEFAULT  CHAR(1) DEFAULT '0' CHECK ( IS_DEFAULT IN ( '0', '1' ) ),
    PRIMARY KEY ( ADDRESS_ID,
                  C_ID ),
    CONSTRAINT CSADDRESSFK FOREIGN KEY ( ADDRESS_ID )
        REFERENCES SHIPPING_DETAILS ( ADDRESS_ID )
            ON DELETE CASCADE,
    CONSTRAINT CSCUSTOMERFK FOREIGN KEY ( C_ID )
        REFERENCES CUSTOMER ( C_ID )
            ON DELETE CASCADE
);

CREATE TABLE PAYMENT_DETAILS (
    PAYMENT_ID      NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    CARD_NO         NUMBER NOT NULL,
    CVV             NUMBER NOT NULL,
    EXPIRY_DATE     DATE NOT NULL,
    BILLING_ADDRESS  VARCHAR(500) NOT NULL,
    PRIMARY KEY ( PAYMENT_ID )
);

CREATE TABLE CUSTOMER_PAYMENT (
    PAYMENT_ID  NUMBER,
    C_ID        CHAR(15),
```

```
    IS_DEFAULT  CHAR(1) DEFAULT '0' CHECK ( IS_DEFAULT IN ( '0', '1' ) ),
    PRIMARY KEY ( C_ID,
                  PAYMENT_ID ),
    CONSTRAINT CPPAYMENTFK FOREIGN KEY ( PAYMENT_ID )
        REFERENCES PAYMENT_DETAILS ( PAYMENT_ID )
            ON DELETE CASCADE,
    CONSTRAINT CPCUSTOMERFK FOREIGN KEY ( C_ID )
        REFERENCES CUSTOMER ( C_ID )
            ON DELETE CASCADE
);

CREATE TABLE SUBSCRIPTION (
    SUBSCRIPTION_ID  NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    C_ID             CHAR(15),
    START_DATE       DATE NOT NULL,
    END_DATE         DATE NOT NULL,
    PRIMARY KEY ( SUBSCRIPTION_ID ),
    CONSTRAINT SCUSTOMERFK FOREIGN KEY ( C_ID )
        REFERENCES CUSTOMER ( C_ID )
            ON DELETE SET NULL
);

CREATE TABLE BUYER (
    BUYER_ID CHAR(15),
    PRIMARY KEY ( BUYER_ID ),
    CONSTRAINT BBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES CUSTOMER ( C_ID )
            ON DELETE CASCADE
);

CREATE TABLE SELLER (
    SELLER_ID    CHAR(15),
    SELLER_TYPE  VARCHAR(20),
    PRIMARY KEY ( SELLER_ID ),
    CONSTRAINT SSELLERFK FOREIGN KEY ( SELLER_ID )
        REFERENCES CUSTOMER ( C_ID )
            ON DELETE CASCADE
);

CREATE TABLE REVIEW (
    REVIEW_ID           NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    BUYER_ID            CHAR(15),
    R_DESC              VARCHAR(500) NOT NULL,
    TITLE               VARCHAR(30) NOT NULL,
    RATING              INT CHECK ( RATING BETWEEN 1 AND 5 ),
    SELLER_PRODUCT_FLAG  CHAR(1) CHECK ( SELLER_PRODUCT_FLAG IN ( 'S', 'P' ) ),
    PRIMARY KEY ( REVIEW_ID ),
    CONSTRAINT RBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES BUYER ( BUYER_ID )
```

```
            ON DELETE SET NULL
);

CREATE TABLE SELLER_REVIEWS (
    SELLER_ID  CHAR(15),
    REVIEW_ID  NUMBER,
    PRIMARY KEY ( SELLER_ID,
                  REVIEW_ID ),
    CONSTRAINT SRSELLERFK FOREIGN KEY ( SELLER_ID )
        REFERENCES SELLER ( SELLER_ID )
            ON DELETE CASCADE,
    CONSTRAINT SRREVIEWFK FOREIGN KEY ( REVIEW_ID )
        REFERENCES REVIEW ( REVIEW_ID )
            ON DELETE CASCADE
);

CREATE TABLE REVIEW_IMAGES (
    REVIEW_ID   NUMBER,
    REVIEW_IMG  CHAR(20),
    PRIMARY KEY ( REVIEW_ID,
                  REVIEW_IMG ),
    CONSTRAINT RIREVIEWFK FOREIGN KEY ( REVIEW_ID )
        REFERENCES REVIEW ( REVIEW_ID )
            ON DELETE CASCADE
);

CREATE TABLE CATEGORY (
    CATEGORY_ID  NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    NAME         VARCHAR(15) NOT NULL,
    C_DESC       VARCHAR(20) NOT NULL,
    PRIMARY KEY ( CATEGORY_ID )
);

CREATE TABLE PRODUCT (
    P_ID         NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    P_NAME       VARCHAR(30) NOT NULL,
    P_DESC       VARCHAR(500) NOT NULL,
    PRICE        NUMBER NOT NULL,
    QTY          INT DEFAULT 0 NOT NULL CHECK ( QTY >= 0 ),
    CATEGORY_ID  NUMBER,
    PRIMARY KEY ( P_ID ),
    CONSTRAINT PCATEGORYFK FOREIGN KEY ( CATEGORY_ID )
        REFERENCES CATEGORY ( CATEGORY_ID )
            ON DELETE SET NULL
);

CREATE TABLE WISHLIST_ITEM (
    PRODUCT_ID  NUMBER,
    BUYER_ID    CHAR(15),
```

```
    PRIMARY KEY ( PRODUCT_ID,
                  BUYER_ID ),
    CONSTRAINT WIPRODUCTFK FOREIGN KEY ( PRODUCT_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE,
    CONSTRAINT WIBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES BUYER ( BUYER_ID )
            ON DELETE CASCADE
);

CREATE TABLE SELLER_PRODUCTS (
    SELLER_ID  CHAR(15),
    P_ID       NUMBER,
    PRIMARY KEY ( SELLER_ID,
                  P_ID ),
    CONSTRAINT SPSELLERFK FOREIGN KEY ( SELLER_ID )
        REFERENCES SELLER ( SELLER_ID )
            ON DELETE CASCADE,
    CONSTRAINT SPPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE
);

CREATE TABLE PRODUCT_REVIEWS (
    P_ID       NUMBER,
    REVIEW_ID  NUMBER,
    PRIMARY KEY ( P_ID,
                  REVIEW_ID ),
    CONSTRAINT PRPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE,
    CONSTRAINT PRREVIEWFK FOREIGN KEY ( REVIEW_ID )
        REFERENCES REVIEW ( REVIEW_ID )
            ON DELETE CASCADE
);

CREATE TABLE PRODUCT_IMAGES (
    P_ID     NUMBER,
    P_IMAGE  CHAR(20),
    PRIMARY KEY ( P_ID,
                  P_IMAGE ),
    CONSTRAINT PIPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE
);

CREATE TABLE DAILY_DEALS (
    P_ID       NUMBER,
    DEAL_DATE  DATE NOT NULL,
    DISCOUNT   NUMBER NOT NULL,
    PRIMARY KEY ( P_ID,
```

```
                DEAL_DATE ),
    CONSTRAINT DDPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE
);

CREATE TABLE CARRIER (
    CARRIER_ID    NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    CARRIER_NAME  VARCHAR(15) NOT NULL,
    PRIMARY KEY ( CARRIER_ID )
);

CREATE TABLE CART (
    CART_ID      NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    BUYER_ID     CHAR(15),
    TOTAL_QTY    INT DEFAULT 0 NOT NULL CHECK ( TOTAL_QTY >= 0 ),
    TOTAL_PRICE  NUMBER DEFAULT 0 NOT NULL,
    PRIMARY KEY ( CART_ID ),
    CONSTRAINT CBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES BUYER ( BUYER_ID )
            ON DELETE CASCADE
);

CREATE TABLE CART_ITEMS (
    CART_ID  NUMBER,
    P_ID     NUMBER,
    QTY      INT DEFAULT 1 NOT NULL CHECK ( QTY >= 0 ),
    PRIMARY KEY ( CART_ID,
                  P_ID ),
    CONSTRAINT CICARTFK FOREIGN KEY ( CART_ID )
        REFERENCES CART ( CART_ID )
            ON DELETE CASCADE,
    CONSTRAINT CIPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE CASCADE
);

CREATE TABLE DISCOUNT (
    DISCOUNT_ID    NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    DISCOUNT_NAME  VARCHAR(20) NOT NULL,
    D_DESC         VARCHAR(100) NOT NULL,
    DISCOUNT_AMT   NUMBER NOT NULL CHECK ( DISCOUNT_AMT >= 0 ),
    PRIMARY KEY ( DISCOUNT_ID )
);

CREATE TABLE ORDERS (
    ORDER_ID     NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
```

```
    BUYER_ID     CHAR(15),
    DISCOUNT_ID  NUMBER,
    PAYMENT_ID   NUMBER,
    ORDER_DATE   DATE NOT NULL,
    PRIMARY KEY ( ORDER_ID ),
    CONSTRAINT OBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES BUYER ( BUYER_ID )
            ON DELETE SET NULL,
    CONSTRAINT OPAYMENTFK FOREIGN KEY ( PAYMENT_ID )
        REFERENCES PAYMENT_DETAILS ( PAYMENT_ID )
            ON DELETE SET NULL,
    CONSTRAINT ODISCOUNTFK FOREIGN KEY ( DISCOUNT_ID )
        REFERENCES DISCOUNT ( DISCOUNT_ID )
            ON DELETE SET NULL
);

CREATE TABLE SHIPMENT (
    SHIPPING_ID          NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    ORDER_ID             NUMBER,
    P_ID                 NUMBER,
    CARRIER_ID           NUMBER,
    SHIPMENT_TYPE        CHAR(2) DEFAULT 'NP' CHECK ( SHIPMENT_TYPE IN ( 'PP',
'NP' ) ),
    STATUS               VARCHAR(10) NOT NULL,
    EST_DELIVERY_DATE    DATE,
    ACTUAL_DELIVERY_DATE  DATE,
    PRIMARY KEY ( SHIPPING_ID ),
    CONSTRAINT SORDERFK FOREIGN KEY ( ORDER_ID )
        REFERENCES ORDERS ( ORDER_ID )
            ON DELETE SET NULL,
    CONSTRAINT SPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
            ON DELETE SET NULL,
    CONSTRAINT SCARRIERFK FOREIGN KEY ( CARRIER_ID )
        REFERENCES CARRIER ( CARRIER_ID )
            ON DELETE SET NULL
);

CREATE TABLE RETURNS (
    RETURN_ID  NUMBER
        GENERATED BY DEFAULT ON NULL AS IDENTITY,
    BUYER_ID   CHAR(15),
    P_ID       NUMBER,
    ORDER_ID   NUMBER,
    PRIMARY KEY ( RETURN_ID ),
    CONSTRAINT RETBUYERFK FOREIGN KEY ( BUYER_ID )
        REFERENCES BUYER ( BUYER_ID )
            ON DELETE SET NULL,
    CONSTRAINT RETPRODUCTFK FOREIGN KEY ( P_ID )
        REFERENCES PRODUCT ( P_ID )
```

```
            ON DELETE SET NULL,
    CONSTRAINT RETORDERFK FOREIGN KEY ( ORDER_ID )
        REFERENCES ORDERS ( ORDER_ID )
            ON DELETE SET NULL
);
```

Relevant Procedures:

Procedure to check whether a discount from the daily deals is applicable when placing an order –

```
CREATE OR REPLACE PROCEDURE APPLY_DAILY_DEALS (
    CART           IN      CART.CART_ID%TYPE,
    O_DATE         IN      ORDERS.ORDER_DATE%TYPE,
    MINIMUM_PRICE  IN      CART.TOTAL_PRICE%TYPE,
    TOTAL          IN OUT  CART.TOTAL_PRICE%TYPE
) AS

    CURSOR PRODUCTS IS
    SELECT
        *
    FROM
        CART_ITEMS
    WHERE
        CART_ID = CART;

    THIS_PRODUCT      PRODUCTS%ROWTYPE;
    DISCOUNT_PERCENT  DAILY_DEALS.DISCOUNT%TYPE;
    ITEM_PRICE        PRODUCT.PRICE%TYPE;
    ITEM_DISCOUNT     NUMBER;
BEGIN
    OPEN PRODUCTS;
    LOOP
        FETCH PRODUCTS INTO THIS_PRODUCT;
        EXIT WHEN ( PRODUCTS%NOTFOUND );
        BEGIN
            SELECT
                DISCOUNT
            INTO DISCOUNT_PERCENT
            FROM
                DAILY_DEALS DD
            WHERE
                DD.P_ID = THIS_PRODUCT.P_ID
                AND DD.DEAL_DATE = O_DATE;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DISCOUNT_PERCENT := 0;
        END;

        SELECT
            PRICE
        INTO ITEM_PRICE
        FROM
            PRODUCT
        WHERE
            P_ID = THIS_PRODUCT.P_ID;
```

```
        FOR IND IN 1..THIS_PRODUCT.QTY LOOP
            ITEM_DISCOUNT := ( DISCOUNT_PERCENT / 100 ) * ITEM_PRICE;
            TOTAL := TOTAL - ITEM_DISCOUNT;
        END LOOP;

        IF TOTAL < MINIMUM_PRICE THEN
            TOTAL := MINIMUM_PRICE;
        END IF;
    END LOOP;

    CLOSE PRODUCTS;
END APPLY_DAILY_DEALS;
```

Relevant Triggers:

Trigger to update the inventory when an order is placed –

```
CREATE OR REPLACE TRIGGER UPDATE_INVENTORY BEFORE
    INSERT ON ORDERS
    FOR EACH ROW
DECLARE
    CURSOR ITEMS IS
    SELECT
        CI.P_ID,
        CI.QTY
    FROM
        CART_ITEMS CI,
        CART
    WHERE
        CI.CART_ID = CART.CART_ID
        AND CART.BUYER_ID = :NEW.BUYER_ID;

    THIS_ITEM  ITEMS%ROWTYPE;
    NEW_QTY    NUMBER := 0;
    OLD_QTY    NUMBER;
BEGIN
    OPEN ITEMS;
    LOOP
        FETCH ITEMS INTO THIS_ITEM;
        EXIT WHEN ( ITEMS%NOTFOUND );
        SELECT
            QTY
        INTO OLD_QTY
        FROM
            PRODUCT
        WHERE
            P_ID = THIS_ITEM.P_ID;

        NEW_QTY := OLD_QTY - THIS_ITEM.QTY;
        IF NEW_QTY < 0 THEN
            RAISE_APPLICATION_ERROR(
                                -20000,
                                'Not enough stock'
            );
        END IF;
        UPDATE PRODUCT
        SET
            QTY = NEW_QTY
        WHERE
            P_ID = THIS_ITEM.P_ID;

    END LOOP;

    CLOSE ITEMS;
```

```
END;
```

Trigger to reset a user's cart after an order is placed –

```
CREATE OR REPLACE TRIGGER EMPTY_CART AFTER
    INSERT ON ORDERS
    FOR EACH ROW
DECLARE
    CID CART.CART_ID%TYPE;
BEGIN
    SELECT
        CART_ID
    INTO CID
    FROM
        CART
    WHERE
        BUYER_ID = :NEW.BUYER_ID;

    DELETE FROM CART_ITEMS
    WHERE
        CART_ID = CID;

    UPDATE CART
    SET
        CART.TOTAL_QTY = 0,
        CART.TOTAL_PRICE = 0
    WHERE
        CART_ID = CID;

END;
```

Trigger to update total quantity and total price of the cart when an item is added –

```
CREATE OR REPLACE TRIGGER UPDATE_CART_DETAILS AFTER
    INSERT ON CART_ITEM S
    FOR EACH ROW
DECLARE
    ITEM_PRICE   PRODUCT.PRICE%TYPE;
    ADDED_PRICE  PRODUCT.PRICE%TYPE;
BEGIN
    UPDATE CART
    SET
        TOTAL_QTY = :NEW.QTY + TOTAL_QTY
    WHERE
        CART_ID = :NEW.CART_ID;

    SELECT
        PRICE
    INTO ITEM_PRICE
    FROM
        PRODUCT
```

```
    WHERE
        P_ID = :NEW.P_ID;

    ADDED_PRICE := ITEM_PRICE * :NEW.QTY;
    UPDATE CART
    SET
        TOTAL_PRICE = TOTAL_PRICE + ADDED_PRICE
    WHERE
        CART_ID = :NEW.CART_ID;

END;
```