ONLINE RESELLING MARKETPLACE

A
Practical Activity Report
Submitted for
DBMS

Submitted by:

NIKHIL SHARMA 102203474

MEHAK SINGLA 102203657

VANSH 102203655

(2CO15)

Computer Science and Engineering Department

TIET, Patiala

TABLE OF CONTENTS

ABSTRACT

An online re-selling site is a platform where users can sell and purchase used items. This requires a well-structured database to manage the information related to users, their ads, and orders.

The following is the requirement analysis of a database design for an online re-selling site based on the implemented schemas.

The first requirement is to store and manage product information. The PRODUCT table contains information about the product, such as the product ID, name, condition, image, category, price, and quantity. The table should be designed to handle multiple images for a single product, and the category should be normalized to a separate table to avoid data redundancy.

The second requirement is to store and manage user profiles. The USERS table contains information about the user, such as the user ID, first name, last name, email, and phone number. The phone number should be stored as a string to allow for international numbers. The table should also include fields for billing and shipping addresses to simplify the checkout process.

The third requirement is to manage orders. The ORDERS table contains information about the order, such as the order ID, product ID, buyer ID, amount, order date, and status ID. The table should be normalized to handle multiple products in a single order, and the status ID should be normalized to a separate table to allow for easy tracking of order status.

The fourth requirement is to manage delivery details. The DELIVERY_DETAILS table contains information about the delivery, such as the order ID, address, and expected arrival date. The table should be linked to the ORDERS table to ensure that the delivery details are accurate and up-to-date.

The fifth requirement is to manage payment information. The CARD_DETAILS table contains information about the user's payment card, such as the card number and bank name. The table should be linked to the USERS table to ensure that the payment information is accurate and up-to-date.

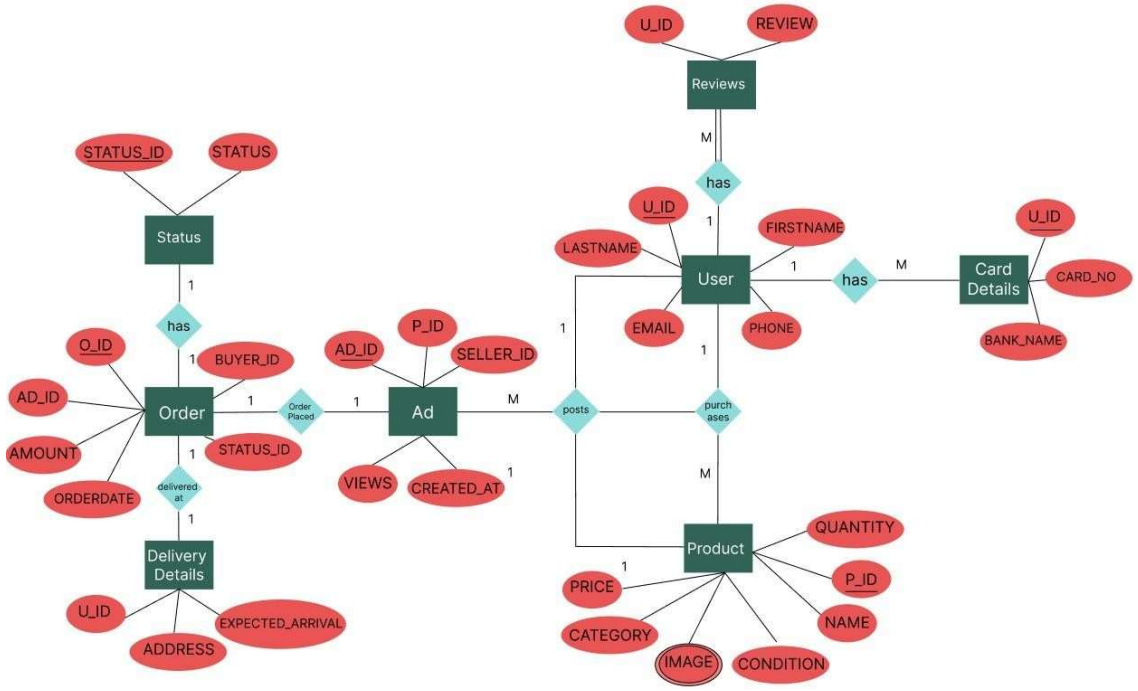
In summary, the database design for an online re-selling site should be able to store and manage product information, user profiles, orders, delivery details, and payment information. The tables should be designed to avoid data redundancy and ensure data accuracy and consistency. The design should also be scalable and flexible to accommodate future growth and changes in the business requirements.

To implement the above schema for an online re-selling site, several functionalities are required. These functionalities include creating ads, deleting ads, placing orders, canceling orders, updating user profiles, displaying delivery status, and displaying available categories. These functionalities require different SQL queries and triggers to be implemented.

A trigger can be implemented to delete an ad when an order is delivered. Similarly, a trigger can be implemented to display the delivery status when it changes. The database must also ensure data integrity by implementing referential integrity constraints to ensure that no data is lost or corrupted.

To make the site user-friendly, the system should be easy to navigate, and search functionalities should be implemented to make it easier for users to find what they are looking for. Additionally, the system should implement secure and durable .Thus, this site requires careful planning and designing of a well-structured database, the implementation of various SQL queries and triggers. It also requires the implementation of security measures to protect users' sensitive information.

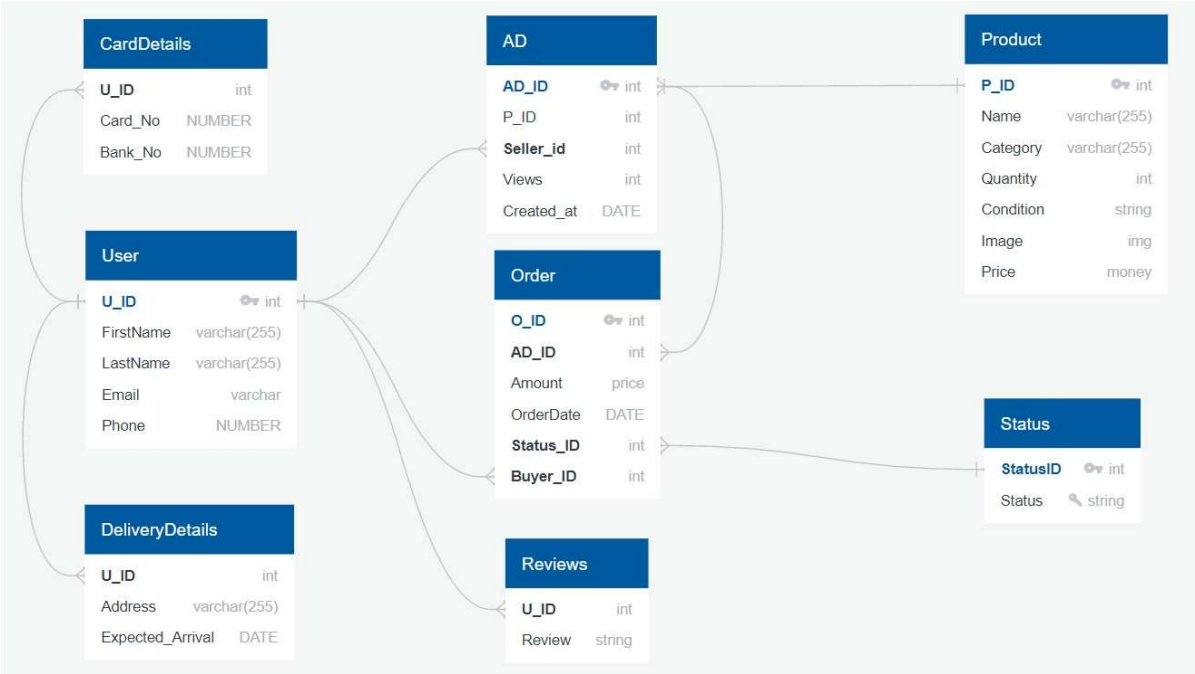
ER DIAGRAM



ASSUMPTIONS:

- A user can purchase many products (1:M)
- A user can sell many products (1:M)
- A user can have many cards (1:M)
- Every order has one set of delivery details (1:1)
- An order can have a single status value(1:1)
- Every order corresponds to a single Ad(1:1)
- A user can have multiple reviews (1:M)

MAPPING OF ERD IN RELATIONAL SCHEMA



TABLES CREATION

```
CREATE TABLE PRODUCT(P_ID INTEGER PRIMARY KEY,
                NAME VARCHAR(20),
                CONDITION VARCHAR(100),
                IMAGE VARCHAR(50),
                CATEGORY VARCHAR(30),
                PRICE  INTEGER,
                QUANTITY INTEGER);



CREATE TABLE USERS (U_ID INTEGER PRIMARY KEY,
   FIRSTNAME VARCHAR(15),
   LASTNAME VARCHAR(15),
   EMAIL VARCHAR(20),
   PHONE INTEGER CHECK (LENGTH(PHONE)=10)
   );



CREATE TABLE ADS(
   AD_ID INTEGER PRIMARY KEY,
   P_ID INTEGER REFERENCES PRODUCT(P_ID),
   SELLER_ID INTEGER REFERENCES USERS(U_ID),
   CREATED_AT DATE,
   VIEWS INTEGER

);


CREATE TABLE ORDERS(O_ID INTEGER PRIMARY KEY,
                AD_ID INTEGER REFERENCES ADS(AD_ID),
                AMOUNT INTEGER,
                ORDERDATE DATE,
                STATUS_ID INTEGER REFERENCES STATUS(STATUS_ID),
                BUYER_ID INTEGER REFERENCES USERS(U_ID));
```

```
CREATE TABLE USER_REVIEWS(
   U_ID INTEGER NOT NULL REFERENCES USERS(U_ID),
   REVIEW VARCHAR(200)
);

CREATE TABLE STATUS(STATUS_ID INTEGER PRIMARY KEY,
                STATUS VARCHAR(100));

CREATE TABLE DELIVERY_DETAILS(
   O_ID INTEGER REFERENCES ORDERS(O_ID),
   ADDRESS VARCHAR(100),
   EXPECTED_ARRIVAL DATE

);

CREATE TABLE CARD_DETAILS(
   U_ID INTEGER REFERENCES USERS(U_ID),
   CARD_NO INTEGER UNIQUE,
   BANK_NAME VARCHAR(200)
);
```

CONSTRAINTS

```
ALTER TABLE USERS MODIFY FIRSTNAME VARCHAR(15) NOT NULL;

ALTER TABLE CARD_DETAILS ADD CONSTRAINT UQ_CARD_NO UNIQUE
(CARD_NO);

ALTER TABLE ADS ADD CONSTRAINT AD_PRODUCT_FK
FOREIGN KEY (P_ID) REFERENCES PRODUCT (P_ID) ON DELETE CASCADE;

ALTER TABLE PRODUCT
ADD CONSTRAINT CHECK_PRICE CHECK (PRICE >= 0);
```

VALUES INSERTION

```
--Product table values
INSERT INTO PRODUCT VALUES(1, 'T-shirt','Good','pic.com','Clothing',299,1);
INSERT INTO PRODUCT VALUES(2, 'Laptop','Like new','abc.com','Electronics',29999,1);
INSERT INTO PRODUCT VALUES(3, 'Levis jeans','Good','ssd.com','Clothing',799,1);
INSERT INTO PRODUCT VALUES(4, 'Lab Coat','Okay','lab.com','Clothing',249,2);
INSERT INTO PRODUCT VALUES(5, 'Bay-Blade','Excellent','toy.com','Toys',199,1);
INSERT INTO PRODUCT VALUES(6, 'Wired Mouse','Good','m.in','Electronics',499,5);
INSERT INTO PRODUCT VALUES(7, 'Keyboard','Nice','k.in','Electronics',1099,1);
INSERT INTO PRODUCT VALUES(8, 'Monitor','Unboxed','hehe.com','Electronics',7999,1);
INSERT INTO PRODUCT VALUES(9, 'Tool-set','Like new','tool.in','Utility',2799,1);
INSERT INTO PRODUCT VALUES(10, 'Let us C++','Year old','book.in','Books',399,1);
INSERT INTO PRODUCT VALUES(11, 'Harry Potter','2 years old','pic.com','Book',799,2);
INSERT INTO PRODUCT VALUES(12, 'Steam
iron','Unboxed','zap.com','Appliance',1299,1);
INSERT INTO PRODUCT VALUES(13, 'Shorts','Like new','ss.com','Clothing',499,2);
INSERT INTO PRODUCT VALUES(14, 'IPhone
8','128GB','apple.com','Electronics',20999,1);
INSERT INTO PRODUCT VALUES(15, 'Bagpack','Good','gen.com','Utiity',999,2);

--User table values
INSERT INTO USERS VALUES(1,'Amit','Kumar','akumar1@gmail.com',9874536271);
INSERT INTO USERS VALUES(2,'Bhavya','Mittal','bmit@yahoo.com',9756381430);
INSERT INTO USERS VALUES(3,'Chirag','Singla','csingla1@thapar.edu',8966745362);
INSERT INTO USERS VALUES(4,'Dhananjay','Singh','dsingh@outlook.com',9867856473);
INSERT INTO USERS VALUES(5,'Eli','Roth','elisgreat1@gmail.com',9987500845);
INSERT INTO USERS VALUES(6,'Farhan','Qureshi','farhan2@yahoo.com',8900065784);
INSERT INTO USERS VALUES(7,'Garima','Arora','arima1@gmail.com',9806444837);
INSERT INTO USERS VALUES(8,'Harpreet','Sodhi','sodhi23@gmail.com',9800756223);
INSERT INTO USERS VALUES(9,'Ishaan','Soin','ishaan@live.com',9800657899);
INSERT INTO USERS VALUES(10,'Jai','Dalmotra','jaydal@yahoo.com',8989076453);
INSERT INTO USERS VALUES(11,'Kalki','Koechlin','kalki@gmail.com',7890453329);
INSERT INTO USERS VALUES(12,'Lakshay','Gupta','gupay1@gmail.com',8769504956);
INSERT INTO USERS VALUES(13,'Manan','Singla','singmanan@yahoo.com',9087789564);
INSERT INTO USERS VALUES(14,'Nupur','Arora','nupur23@gmail.com',7895085647);
INSERT INTO USERS VALUES(15,'Ojas','Gautam','ojasg@gmail.com',8907564783);
INSERT INTO USERS VALUES(16,'Paras','Gupta','pgu1@thapar.edu',8721314156);
INSERT INTO USERS VALUES(17,'Quill','Minar','delhie@gmail.com',6789567483);
INSERT INTO USERS VALUES(18,'Rajesh','Mehta','rajesh@gmail.com',8975647392);
```

```
INSERT INTO USERS VALUES(20,'Dhananjay','Singh','singg@gmail.com',8907567498);

--AD Table Values
INSERT INTO ADS VALUES(101,1,1,TO_DATE('12-01-2023','DD-MM-RR'),5);
INSERT INTO ADS VALUES(102,2,2,TO_DATE('24-02-2023','DD-MM-RR'),10);
INSERT INTO ADS VALUES(103,3,4,TO_DATE('16-01-2023','DD-MM-RR'),11);
INSERT INTO ADS VALUES(104,4,3,TO_DATE('22-03-2023','DD-MM-RR'),15);
INSERT INTO ADS VALUES(105,5,9,TO_DATE('08-04-2023','DD-MM-RR'),4);
INSERT INTO ADS VALUES(106,6,5,TO_DATE('12-02-2023','DD-MM-RR'),56);
INSERT INTO ADS VALUES(107,7,1,TO_DATE('11-01-2023','DD-MM-RR'),6);
INSERT INTO ADS VALUES(108,8,10,TO_DATE('12-01-2023','DD-MM-RR'),5);
INSERT INTO ADS VALUES(109,9,12,TO_DATE('16-04-2023','DD-MM-RR'),10);
INSERT INTO ADS VALUES(110,10,15,TO_DATE('18-03-2023','DD-MM-RR'),23);
INSERT INTO ADS VALUES(111,11,8,TO_DATE('29-03-2023','DD-MM-RR'),12);
INSERT INTO ADS VALUES(112,12,7,TO_DATE('07-04-2023','DD-MM-RR'),17);

--STATUS TABLE VALUES
INSERT INTO STATUS VALUES(691,'DELIVERED');
INSERT INTO STATUS VALUES(692,'PLACED');
INSERT INTO STATUS VALUES(693,'IN-TRANSIT');
INSERT INTO STATUS VALUES(694,'CANCELLED');

--ORDER TABLE VALUES
INSERT INTO ORDERS
VALUES(1001,101,299,TO_DATE('11-02-2024','DD-MM-RR'),691,1);
INSERT INTO ORDERS
VALUES(1002,112,1299,TO_DATE('23-03-2024','DD-MM-RR'),692,3);
INSERT INTO ORDERS
VALUES(1003,109,2799,TO_DATE('02-04-2024','DD-MM-RR'),693,5);
INSERT INTO ORDERS
VALUES(1004,104,249,TO_DATE('14-01-2024','DD-MM-RR'),694,4);
INSERT INTO ORDERS
VALUES(1005,105,199,TO_DATE('09-03-2024','DD-MM-RR'),692,12);
INSERT INTO ORDERS
VALUES(1006,107,1099,TO_DATE('22-02-2024','DD-MM-RR'),693,11);
INSERT INTO ORDERS
VALUES(1007,108,7999,TO_DATE('20-01-2024','DD-MM-RR'),694,16);
INSERT INTO ORDERS
VALUES(1008,110,399,TO_DATE('06-04-2024','DD-MM-RR'),691,8);
INSERT INTO ORDERS
VALUES(1009,103,799,TO_DATE('11-03-2024','DD-MM-RR'),691,5);
INSERT INTO ORDERS
VALUES(1010,106,499,TO_DATE('21-01-2024','DD-MM-RR'),694,11);
```

```
--USER_REVIEWS TABLE VALUES
INSERT INTO USER_REVIEWS VALUES(1,'Really satisfied');
INSERT INTO USER_REVIEWS VALUES(4,'Could not have been better');
INSERT INTO USER_REVIEWS VALUES(5,'Mast ekdum');
INSERT INTO USER_REVIEWS VALUES(17,'Worst Experience');
INSERT INTO USER_REVIEWS VALUES(11,'That is what she said');
INSERT INTO USER_REVIEWS VALUES(12,'How the turntables');
INSERT INTO USER_REVIEWS VALUES(8,'Bazzinga');
INSERT INTO USER_REVIEWS VALUES(15,'Did not like it.');
INSERT INTO USER_REVIEWS VALUES(9,'Meh');
INSERT INTO USER_REVIEWS VALUES(10,'Nice');

--DELIVERY_DETAILS TABLE VALUES
INSERT INTO DELIVERY_DETAILS VALUES (1001,'331, Model
Town',TO_DATE('12-03-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1002,'2048,
Bilaspur',TO_DATE('01-05-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1003,'21,
Chandigarh',TO_DATE('23-03-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1004,'46,
Patiala',TO_DATE('11-04-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1005,'678, Punjabi
Bagh',TO_DATE('09-02-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1006,'564,
Pitampura',TO_DATE('18-01-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1007,'23,
Rohini',TO_DATE('16-03-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1008,'36, China
Town',TO_DATE('07-04-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1009,'12/24 Karol
Bagh',TO_DATE('24-03-2024','DD-MM-RR'));
INSERT INTO DELIVERY_DETAILS VALUES (1010,'221B, Baker
Street',TO_DATE('16-03-2024','DD-MM-RR'));

--CARD_DETAILS TABLE VALUES
INSERT INTO CARD_DETAILS VALUES(1,4785930204,'HDFC');
INSERT INTO CARD_DETAILS VALUES(2,4030985219,'AXIS');
INSERT INTO CARD_DETAILS VALUES(9,4502956784,'SBI');
INSERT INTO CARD_DETAILS VALUES(7,4009675894,'ICICI');
INSERT INTO CARD_DETAILS VALUES(6,4096785940,'HDFC');
INSERT INTO CARD_DETAILS VALUES(12,4330578594,'PNB');
INSERT INTO CARD_DETAILS VALUES(15,4567678421,'CITI');
INSERT INTO CARD_DETAILS VALUES(18,4690983422,'KOTAK');
INSERT INTO CARD_DETAILS VALUES(20,5890327560,'SBI');
```

PROCEDURES

-- CREATE AD

```
CREATE OR REPLACE PROCEDURE CREATE_AD (p_product_id IN INTEGER,
   p_seller_id IN INTEGER
) AS l_ad_id INTEGER;
BEGIN
   -- LET AD_ID=19001
   INSERT INTO ADS (AD_ID, P_ID, SELLER_ID, CREATED_AT, VIEWS)
   VALUES (19001, p_product_id, p_seller_id, SYSDATE, 0)
   RETURNING AD_ID INTO l_ad_id;
   DBMS_OUTPUT.PUT_LINE('New ad created with ID ' || l_ad_id);
END;
/
BEGIN
CREATE_AD(14,9);
END;
/
```

--DELETE AD

```
CREATE OR REPLACE PROCEDURE delete_ad(
  p_ad_id IN INTEGER ) IS
BEGIN
   DELETE FROM ADS WHERE AD_ID = p_ad_id;
END;
/
BEGIN
 delete_ad(19001);
END;
```

```
/
```

--PLACE ORDER

```
CREATE OR REPLACE PROCEDURE place_order(      p_ad_id IN ADS.AD_ID%TYPE,
   p_amount IN ORDERS.AMOUNT%TYPE,
   p_buyer_id IN ORDERS.BUYER_ID%TYPE
)
AS
   v_status_id STATUS.STATUS_ID%TYPE;
   v_order_id ORDERS.O_ID%TYPE;
BEGIN
   -- INSERT O_ID =2424
        SELECT STATUS_ID INTO v_status_id FROM STATUS WHERE STATUS =
'PLACED';


      INSERT INTO ORDERS (O_ID, AD_ID, AMOUNT, ORDERDATE, STATUS_ID,
BUYER_ID)
   VALUES (2424, p_ad_id, p_amount, SYSDATE, v_status_id, p_buyer_id)
   RETURNING O_ID INTO v_order_id;


   DBMS_OUTPUT.PUT_LINE('Order placed successfully. Order ID: ' || v_order_id);
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error placing order: ' || SQLERRM);
END;
/
   BEGIN
place_order(104,5700,5);
   END;
/
```

--CANCEL ORDER

```
CREATE OR REPLACE PROCEDURE cancel_order(
   p_order_id IN INTEGER
) IS
BEGIN
      UPDATE ORDERS SET STATUS_ID = (select STATUS_ID from STATUS where STATUS='CANCELLED') WHERE O_ID = p_order_id;
END;
/
   BEGIN
   cancel_order(2424);
   END;
/
```

--DISPLAY DELIVERY STATUS

```
CREATE OR REPLACE FUNCTION delivery_status(
   p_order_id IN INTEGER
) RETURN VARCHAR2 IS
   p_status VARCHAR2(100);
BEGIN
     SELECT STATUS INTO p_status FROM STATUS WHERE STATUS_ID = (SELECT STATUS_ID FROM ORDERS WHERE O_ID = p_order_id);
   RETURN p_status;
END;
/
```

--DISPLAY AVAILABLE CATEGORIES

```
CREATE OR REPLACE PROCEDURE available_categories IS
   c PRODUCT.CATEGORY%TYPE;
  cursor p_category is SELECT DISTINCT CATEGORY FROM PRODUCT;
   c_row p_category%ROWTYPE;

BEGIN
```

```
    OPEN p_category;

LOOP
   FETCH  p_category INTO c_row;
    EXIT WHEN p_category%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE(c_row.CATEGORY);
END LOOP;
   CLOSE p_category;

END;
/

BEGIN
   available_categories;
END;
/
```

TRIGGERS

--TRIGGER INCREASE VIEWCOUNT ON DISPLAY OF AD

```
CREATE OR REPLACE TRIGGER increase_viewcount
AFTER UPDATE OF VIEWS ON ADS
FOR EACH ROW
BEGIN
    UPDATE ADS SET VIEWS = VIEWS + 1 WHERE P_ID = :NEW.P_ID;END;
```

--DISPLAY TO USER WHEN STATUS CHANGES

```
CREATE OR REPLACE TRIGGER display_delivery_status
AFTER UPDATE OF STATUS_ID ON ORDERS
FOR EACH ROW
```

```
   DECLARE
   NS STATUS.STATUS%TYPE;
BEGIN
  IF :OLD.STATUS_ID <> :NEW.STATUS_ID THEN
   SELECT STATUS INTO NS FROM STATUS WHERE STATUS_ID =
:NEW.STATUS_ID;
   DBMS_OUTPUT.PUT_LINE('Delivery status changed to ' || NS);
  END IF;
END;
/

SELECT * FROM ORDERS;
UPDATE ORDERS SET STATUS_ID = 693 WHERE O_ID = 2;
```

NORMALISATION OF SCHEMAS

Normalization is the process of organizing data in a database to eliminate data redundancy and improve data integrity. The above schemas are not fully normalized, and there are several functional dependencies that can be identified.

The steps of normalizations that can be done are:

Step 1: Eliminating redundant data in USERS table

Create a new table to store phone numbers and addresses of users, since users can have multiple phone numbers and addresses. The new table will have a foreign key to the USERS table.

Step 2: Normalizing PRODUCT table

Create a new table for product conditions with a primary key of condition ID and a description of the condition.

Create a new table for product images with a primary key of image ID and a path to the image file.

Create a new table for categories with a primary key of category ID and a category name. The PRODUCT table will have a foreign key to the categories table.

Step 3: Normalizing ADS table

The ADS table has a foreign key to the PRODUCT table. However, since multiple ads can have the same product, create a new table to store unique product instances with a primary key of product instance ID and a foreign key to the PRODUCT table. The ADS table will have a foreign key to the product instances table.

Step 4: Normalizing ORDERS table

The ORDERS table has a foreign key to the ADS table. However, since multiple orders can have the same ad, create a new table to store unique ad instances with a primary key of ad instance ID and a foreign key to the ADS table. The ORDERS table will have a foreign key to the ad instances table.

The ORDERS table has a foreign key to the USERS table. However, since users can have multiple orders, create a new table to store order instances with a primary key of order instance ID and foreign keys to the ORDERS and USERS tables.

Functional dependencies in the fully normalized schema :

USERS:

U_ID -> FIRSTNAME, LASTNAME, EMAIL

U_ID -> {PHONE_NUMBER}

U_ID -> {ADDRESS}

PRODUCT:

P_ID -> NAME, PRICE, QUANTITY

P_ID -> CONDITION_ID, IMAGE_ID, CATEGORY_ID

CONDITION:

CONDITION_ID -> CONDITION_DESCRIPTION

IMAGE:

IMAGE_ID -> IMAGE_PATH

CATEGORY:

CATEGORY_ID -> CATEGORY_NAME

ADS:

AD_ID -> P_INSTANCE_ID, SELLER_ID, CREATED_AT, VIEWS

P_INSTANCE_ID -> P_ID

ORDERS:

O_ID -> AD_INSTANCE_ID, AMOUNT, ORDERDATE, STATUS_ID, BUYER_ID

AD_INSTANCE_ID -> AD_ID

BUYER_ID -> U_ID

DELIVERY_DETAILS:

O_ID -> {ADDRESS, EXPECTED_ARRIVAL}

STATUS:

STATUS_ID -> STATUS_DESCRIPTION

CARD_DETAILS:

U_ID -> CARD_NO, BANK_NAME

USER_REVIEWS:

U_ID -> REVIEW

QUERY OUTPUTS

Create ad with P_ID=14 and SELLER_ID=9

```
Statement processed.
New ad created with ID 19001
```

AD_ID	P_ID	SELLER_ID	CREATED_AT	VIEWS
101	1	1	12-JAN-24	5
102	2	2	24-FEB-24	10
103	3	4	16-JAN-24	11
104	4	3	22-MAR-24	15
105	5	9	08-APR-24	4
106	6	5	12-FEB-24	56
107	7	1	11-JAN-24	6
108	8	10	12-JAN-24	5
109	9	12	16-APR-24	10
110	10	15	18-MAR-24	23
111	11	8	29-MAR-24	12
112	12	7	07-APR-24	17
19001	14	9	05-MAY-24	0

Delete ad with AD_ID=19001

AD_ID	P_ID	SELLER_ID	CREATED_AT	VIEWS
101	1	1	12-JAN-24	5
102	2	2	24-FEB-24	10
103	3	4	16-JAN-24	11
104	4	3	22-MAR-24	15
105	5	9	08-APR-24	4
106	6	5	12-FEB-24	56
107	7	1	11-JAN-24	6
108	8	10	12-JAN-24	5
109	9	12	16-APR-24	10
110	10	15	18-MAR-24	23
111	11	8	29-MAR-24	12
112	12	7	07-APR-24	17

Place order for AD_ID=104 ,AMOUNT=5700, BUYER_ID=5

```
Procedure created.


Statement processed.
Order placed successfully. Order ID: 2424
```

CANCEL ORDER WITH O_ID=2424

```
2  SELECT O_ID, STATUS FROM ORDERS,STATUS WHERE ORDERS.STATUS_ID=STATUS.STATUS_ID;
```

O_ID	STATUS
1001	DELIVERED
1002	PLACED
1003	IN-TRANSIT
1004	CANCELLED
1005	PLACED
1006	IN-TRANSIT
1007	CANCELLED
1008	DELIVERED
1009	DELIVERED
1010	CANCELLED
2424	CANCELLED

DELIVERY STATUS OF O_ID=1003

```
12  BEGIN
13    DBMS_OUTPUT.PUT_LINE(delivery_status(1003));
14  END;
15  /

Function created.

Statement processed.
IN-TRANSIT
```

DISPLAY AVAILABLE CATEGORIES

```
20
21  BEGIN
22     available_categories;
23  END;
24  /

Procedure created.

Statement processed.
Electronics
Book
Utiity
Books
Clothing
Toys
Utility
Appliance
```