DBS Online Marketplace

MIT MANIPAL

DBS

Preface

Dear Reader,

It is with great pleasure that we present this project to you. Through this project, we aim to justify the implemenation of SQL by creating an online marketplace.

We have spent countless hours researching and analyzing the information presented in this document. We would like to express our gratitude to Ms. Sucharitha Shetty, for guiding us through the subject and making us capable enough to complete the project.

Thank you for taking the time to read this project.

Abstract

The online marketplace is a platform designed to facilitate transactions between buyers and sellers. The platform consists of several tables, including the "customer" table, which stores information about registered customers, such as their name, email, phone number, and address. The "seller" table stores information about registered sellers, including their name, phone number, and email address. The "product" table stores information about the products listed for sale on the platform, including the product name, type, color, quantity, description, cost, and the ID of the seller who listed the product. The "purchase" table tracks information about each purchase made through the platform, including the purchase date and the ID of the customer who made the purchase. The "product_seller" table establishes a many-to-many relationship between products and sellers, allowing multiple sellers to list the same product. The "purchase_item" table stores information about the items purchased in each transaction, including the purchase ID, the ID of the product being purchased, the ID of the seller who listed the product, and the quantity purchased. The platform uses triggers to enforce business rules, such as preventing the deletion of sellers with listed products. Overall, the online marketplace provides a streamlined platform for buying and selling products online.

TABLE OF CONTENTS

TABLE OF CONTENTS

Overview

An online marketplace database is a platform that allows multiple sellers to list and sell their products to potential customers. The database serves as a central repository of information about the sellers, their products or services, and their pricing, which can be accessed by customers looking to make a purchase.

The database should have a well-designed schema to store all the necessary information about the sellers, as well as the customers.

Tables Used

The following set of tables provides a basic structure for storing and managing data related to ecommerce transactions, including user information, product information, order and payment details, and product reviews. **All the tables are normalised into 3NF.**

Customer

This table includes personal information of the customers of the online marketplace.

Seller

This table has been created to store information related to the seller.

Products

This table contains information about the products available in the ecommerce system, some of them including their name, description,quantity, etc.

Purchase

This table contains information about the when the product was purchased and what product was purchased, by the customer.

Tables Used

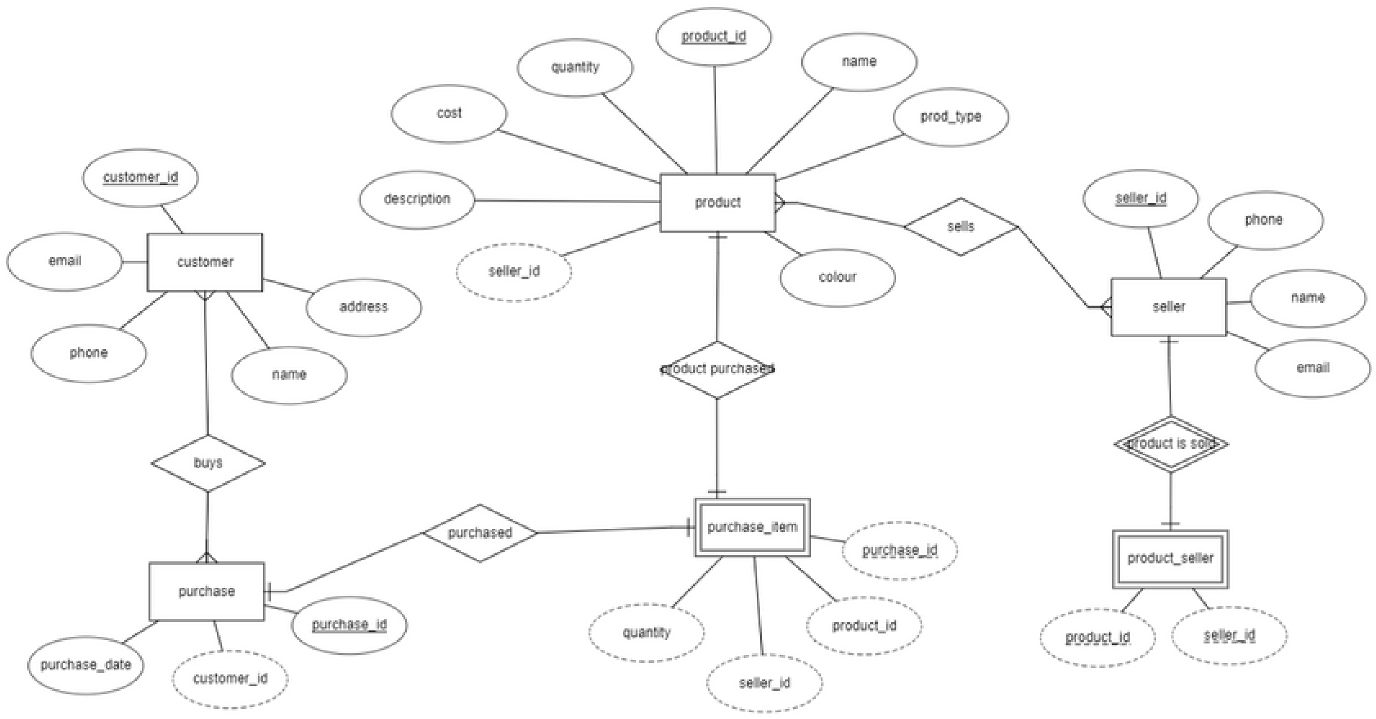
Product_Seller

This table stores information about what product is being sold by the seller.

Purchase_Item

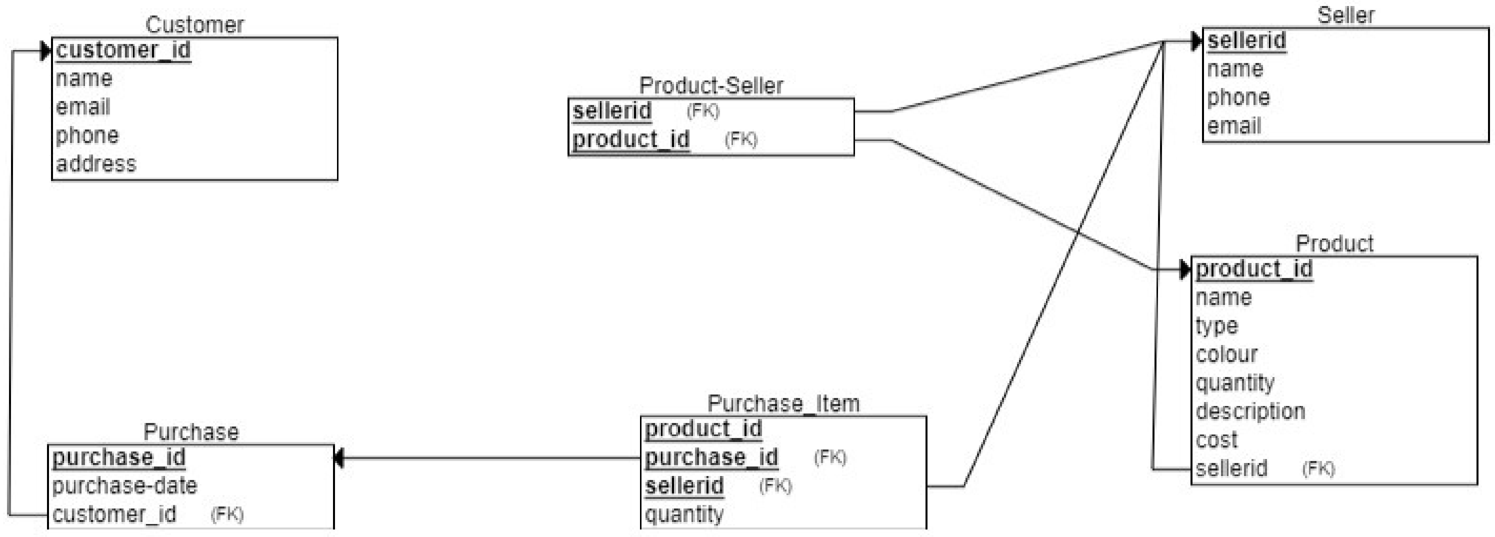
This table is used to store information related to the item purchased

ER Diagram



product_id
quantity
name
cost
prod_type
description
product
sells
seller_id
phone
seller_id
colour
customer_id
email
customer
address
seller
name
phone
name
email
product purchased
product is sold
buys
purchased
purchase_item
purchase_id
product_seller
purchase
purchase_id
quantity
product_id
product_id
seller_id
purchase_date
customer_id
seller_id

Relational Schema



Creation of Tables

```
CREATE TABLE CUSTOMER (
CUSTOMER_ID NUMBER(8) GENERATED ALWAYS AS IDENTITY (START
WITH 1000) PRIMARY KEY,
NAME VARCHAR(45),
EMAIL VARCHAR(45),
PHONE NUMBER(10) UNIQUE,
ADDRESS VARCHAR(100)
);


CREATE TABLE SELLER(
SELLERID NUMBER(8) GENERATED ALWAYS AS IDENTITY (START WITH
2000) PRIMARY KEY,
NAME VARCHAR(20),
PHONE NUMBER(10) UNIQUE,
EMAIL VARCHAR(30));


CREATE TABLE PRODUCT(
PRODUCT_ID NUMBER(8) GENERATED ALWAYS AS IDENTITY (START
WITH 3000) PRIMARY KEY,
NAME VARCHAR(20),
PROD_TYPE VARCHAR(20),
COLOUR VARCHAR(20),
QUANTITY NUMBER(10) NOT NULL,
DESCRIPTION VARCHAR(50),
COST NUMBER(6) NOT NULL,
SELLER_ID NUMBER(8),
FOREIGN KEY(SELLER_ID) REFERENCES SELLER ON DELETE CASCADE);
```

Creation of Tables

```
CREATE TABLE PURCHASE (
PURCHASE_ID NUMBER(10) GENERATED ALWAYS AS IDENTITY (START
WITH 5000) PRIMARY KEY,
PURCHASE_DATE DATE,
CUSTOMER_ID NUMBER(8),
CONSTRAINT FK_CUSTOMER_ID
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER ON DELETE
CASCADE);
```

```
 CREATE TABLE PRODUCT_SELLER(
SELLER_ID NUMBER(8),
PRODUCT_ID NUMBER(8),
PRIMARY KEY(SELLER_ID,PRODUCT_ID),
FOREIGN KEY (SELLER_ID) REFERENCES SELLER ON DELETE CASCADE,
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT ON DELETE
CASCADE);
```

```
CREATE TABLE PURCHASE_ITEM (
    PURCHASE_ID NUMBER(10),
    PRODUCT_ID NUMBER(8),
    SELLER_ID NUMBER(8),
    QUANTITY NUMBER(10),
    PRIMARY KEY (PURCHASE_ID, PRODUCT_ID, SELLER_ID),
    CONSTRAINT FK_PURCHASE_ID
        FOREIGN KEY (PURCHASE_ID) REFERENCES PURCHASE ON
DELETE CASCADE);
```

Insertion of Values

Customer

```
INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('JOHN DOE', 'JOHNDOE@EXAMPLE.COM', 1234567890, '123 MAIN ST,
ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS)
VALUES('JANE SMITH', 'JANESMITH@EXAMPLE.COM', 2345678901, '456
HIGH ST, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS)
VALUES('BOB JOHNSON', 'BOBJOHNSON@EXAMPLE.COM',
3456789012, '789 MAPLE AVE, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('ALICE WILLIAMS', 'ALICEWILLIAMS@EXAMPLE.COM', 4567890123, '321
OAK ST, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('MIKE BROWN', 'MIKEBROWN@EXAMPLE.COM', 5678901234, '654
ELM ST, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('SARA DAVIS', 'SARADAVIS@EXAMPLE.COM', 6789012345, '987 BIRCH
RD, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('TOM WILSON', 'TOMWILSON@EXAMPLE.COM', 7890123456, '543
PINE ST, ANYTOWN USA');
```

Insertion of Values

```
INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('EMILY TAYLOR', 'EMILYTAYLOR@EXAMPLE.COM', 8901234567, '876
CEDAR AVE, ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('JACK LEE', 'JACKLEE@EXAMPLE.COM', 9012345678, '210 SPRUCE ST,
ANYTOWN USA');

INSERT INTO CUSTOMER (NAME, EMAIL, PHONE, ADDRESS) VALUES
('LUCY CHEN', 'LUCYCHEN@EXAMPLE.COM', 1231231234, '333 CHERRY
LN, ANYTOWN USA');
```

Seller

```
INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('ABC STORE',
'5551234567', 'ABCSTORE@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('XYZ STORE',
'8889990000', 'XYZSTORE@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('SUPERMART',
'1112223333', 'SUPERMART@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('AWESOME
DEALS', '6667778888', 'AWESOMEDEALS@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('AMAZING
PRODUCTS', '4445556666', 'AMAZINGPRODUCTS@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('GLOBAL
ENTERPRISES', '7778889999', 'GLOBALENTERPRISES@EXAMPLE.COM');
```

Insertion of Values

```
INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('BEST BUYS',
'3334445555', 'BESTBUYS@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('THE MEGA
STORE', '2223334444', 'THEMEGASTORE@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('TOP BRANDS',
'9876543210', 'TOPBRANDS@EXAMPLE.COM');

INSERT INTO SELLER (NAME, PHONE, EMAIL) VALUES ('PREMIUM
PRODUCTS', '1234567890', 'PREMIUMPRODUCTS@EXAMPLE.COM');
```

Product

```
INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('DUMBBELL SET', 'FITNESS',
'BLACK', 10, 'ADJUSTABLE WEIGHT RANGE', 399.99, 2000);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('T-SHIRT', 'CLOTHING', 'RED',
100, 'COTTON BLEND MATERIAL', 20.99, 2001);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('BACKPACK',
'ACCESSORIES', 'BLACK', 50, 'WATER-RESISTANT MATERIAL', 49.99,
2002);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID)VALUES ('SNEAKERS', 'SHOES',
'WHITE', 75, 'BREATHABLE MESH DESIGN', 79.99, 2003);
```

Insertion of Values

```
INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('SMARTWATCH',
'ELECTRONICS', 'SILVER', 25, 'FITNESS TRACKING FEATURES', 199.99,
2004);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('YOGA MAT', 'SPORTS',
'PURPLE', 30, 'ECO-FRIENDLY MATERIAL', 29.99, 2005);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('LAPTOP', 'COMPUTERS',
'SPACE GRAY', 20, 'INTEL CORE I7 PROCESSOR', 1199.99, 2006);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('GUITAR', 'MUSICAL
INSTRUMENTS', 'NATURAL', 15, 'ACOUSTIC WITH NYLON STRINGS',
299.99, 2007);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('BLENDER', 'KITCHEN
APPLIANCES', 'BLACK', 40, 'POWERFUL 1000W MOTOR', 79.99, 2008);

INSERT INTO PRODUCT (NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID) VALUES ('PENCIL SET', 'ART
SUPPLIES', 'ASSORTED', 60, 'HB AND 2B PENCILS', 14.99, 2009);
```

Insertion of Values

Product_Seller

```
INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2000, 3000);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2001, 3001);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2002, 3002);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2003, 3003);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2004, 3004);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2005, 3005);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2006, 3006);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2007, 3007);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2008, 3008);

INSERT INTO PRODUCT_SELLER (SELLER_ID, PRODUCT_ID) VALUES
(2009, 3009);
```

Queries

Number of products listed by each seller and their average cost

```
SELECT S.NAME, COUNT(P.PRODUCT_ID) AS NUM_PRODUCTS,
AVG(P.COST) AS AVG_COST
FROM SELLER S
LEFT JOIN PRODUCT P ON S.SELLERID = P.SELLER_ID
GROUP BY S.NAME;
```

View created to display orders made by particular customer

```
CREATE VIEW CUSTOMER_PURCHASE AS
SELECT P.PURCHASE_ID, P.PURCHASE_DATE, PR.NAME AS
PRODUCT_NAME, PI.QUANTITY, PR.COST * PI.QUANTITY AS
TOTAL_COST
FROM PURCHASE P
JOIN PURCHASE_ITEM PI ON P.PURCHASE_ID = PI.PURCHASE_ID
JOIN PRODUCT PR ON PI.PRODUCT_ID = PR.PRODUCT_ID
WHERE P.CUSTOMER_ID =1000
ORDER BY P.PURCHASE_DATE DESC;
```

Queries

Get the total revenue generated by each seller in the current month

```
SELECT S.NAME, SUM(PI.QUANTITY * P.COST) AS REVENUE
FROM SELLER S
INNER JOIN PRODUCT P ON S.SELLERID = P.SELLER_ID
INNER JOIN PRODUCT_SELLER PS ON P.PRODUCT_ID =
PS.PRODUCT_ID
INNER JOIN PURCHASE_ITEM PI ON PS.SELLER_ID = PI.SELLER_ID AND
PS.PRODUCT_ID = PI.PRODUCT_ID
INNER JOIN PURCHASE PU ON PI.PURCHASE_ID = PU.PURCHASE_ID
WHERE EXTRACT(MONTH FROM PU.PURCHASE_DATE) =
EXTRACT(MONTH FROM SYSDATE)
GROUP BY S.NAME;
```

Retrieve the list of all customers who have made a purchase in the last 30 days

```
SELECT DISTINCT C.NAME, C.EMAIL, C.PHONE, C.ADDRESS
FROM CUSTOMER C
INNER JOIN PURCHASE P ON C.CUSTOMER_ID = P.CUSTOMER_ID
WHERE P.PURCHASE_DATE >= SYSDATE - 30;
```

Queries

Products purchased on a particular date

```
SELECT P.PURCHASE_ID, P.PURCHASE_DATE, C.NAME AS
CUSTOMER_NAME, C.EMAIL AS CUSTOMER_EMAIL, C.PHONE AS
CUSTOMER_PHONE, C.ADDRESS AS CUSTOMER_ADDRESS,
SUM(PI.QUANTITY * PRO.COST) AS TOTAL_COST
FROM PURCHASE P
INNER JOIN CUSTOMER C ON P.CUSTOMER_ID = C.CUSTOMER_ID
INNER JOIN PURCHASE_ITEM PI ON P.PURCHASE_ID =
PI.PURCHASE_ID
INNER JOIN PRODUCT PRO ON PI.PRODUCT_ID=PRO.PRODUCT_ID
WHERE P.PURCHASE_DATE = TO_DATE('11-05-2023', 'DD-MM-YYYY')
GROUP BY P.PURCHASE_ID, P.PURCHASE_DATE, C.NAME, C.EMAIL,
C.PHONE, C.ADDRESS;
```

List all products in stock

```
SELECT DISTINCT C.NAME, C.EMAIL, C.PHONE, C.ADDRESS
FROM CUSTOMER C
INNER JOIN PURCHASE P ON C.CUSTOMER_ID = P.CUSTOMER_ID
WHERE P.PURCHASE_DATE >= SYSDATE - 30;
```

Queries

Purchases made by customer

```
CREATE VIEW V1 AS
SELECT P.PURCHASE_ID, P.PURCHASE_DATE, PR.NAME AS
PRODUCT_NAME, PI.QUANTITY, PR.COST * PI.QUANTITY AS
TOTAL_COST
FROM PURCHASE P
JOIN PURCHASE_ITEM PI ON P.PURCHASE_ID = PI.PURCHASE_ID
JOIN PRODUCT PR ON PI.PRODUCT_ID = PR.PRODUCT_ID
WHERE P.CUSTOMER_ID =1000
ORDER BY P.PURCHASE_DATE DESC;
```

List all the products sold by a specific seller

```
SELECT P.PRODUCT_ID, P.NAME, P.PROD_TYPE, P.COLOUR,
P.QUANTITY, P.DESCRIPTION, P.COST
FROM PRODUCT P
INNER JOIN SELLER S ON P.SELLER_ID = S.SELLERID
WHERE S.SELLERID=2001;
```

Retrieve the total cost of a particular purchase

```
SELECT SUM(QUANTITY * COST) AS TOTAL_COST FROM
PURCHASE_ITEM JOIN PRODUCT ON PURCHASE_ITEM.PRODUCT_ID =
PRODUCT.PRODUCT_ID WHERE
PURCHASE_ITEM.PURCHASE_ID=5000;
```

Queries

Retrieve the top 10 products by quantity sold

```
SELECT PRODUCT.PRODUCT_ID, NAME,
SUM(PURCHASE_ITEM.QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM PURCHASE_ITEM JOIN PRODUCT ON
PURCHASE_ITEM.PRODUCT_ID = PRODUCT.PRODUCT_ID
GROUP BY PRODUCT.PRODUCT_ID, NAME
ORDER BY TOTAL_QUANTITY_SOLD DESC
FETCH FIRST 10 ROWS ONLY;
```

Procedures

To search for a product

```
CREATE OR REPLACE PROCEDURE SHOW_PRODUCT_DETAILS
(P_NAME VARCHAR)
IS
BEGIN
  FOR R IN (SELECT P.PRODUCT_ID, P.NAME, P.PROD_TYPE,
P.COLOUR, P.QUANTITY, P.DESCRIPTION, P.COST
       FROM PRODUCT P
       WHERE UPPER(P.NAME) LIKE UPPER('%' || P_NAME || '%'))
  LOOP
    DBMS_OUTPUT.PUT_LINE('PRODUCT ID: ' || R.PRODUCT_ID);
    DBMS_OUTPUT.PUT_LINE('NAME: ' || R.NAME);
    DBMS_OUTPUT.PUT_LINE('TYPE: ' || R.PROD_TYPE);
    DBMS_OUTPUT.PUT_LINE('COLOUR: ' || R.COLOUR);
    DBMS_OUTPUT.PUT_LINE('QUANTITY: ' || R.QUANTITY);
    DBMS_OUTPUT.PUT_LINE('DESCRIPTION: ' || R.DESCRIPTION);
    DBMS_OUTPUT.PUT_LINE('COST: ' || R.COST);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
  END LOOP;
END;
/
```

Procedures

To search for products by type

```
CREATE OR REPLACE PROCEDURE SHOW_PRODUCTS_BY_TYPE
(P_TYPE IN VARCHAR2)
IS
BEGIN
 FOR REC IN (
  SELECT *
  FROM PRODUCT
  WHERE UPPER(PROD_TYPE) LIKE UPPER('%' || P_TYPE || '%')
 ) LOOP
 DBMS_OUTPUT.PUT_LINE('PRODUCT ID: ' || REC.PRODUCT_ID);
   DBMS_OUTPUT.PUT_LINE(' NAME: ' || REC.NAME);
   DBMS_OUTPUT.PUT_LINE(' TYPE: ' || REC.PROD_TYPE);
   DBMS_OUTPUT.PUT_LINE(' COLOUR: ' || REC.COLOUR);
   DBMS_OUTPUT.PUT_LINE(' QUANTITY: ' || REC.QUANTITY);
   DBMS_OUTPUT.PUT_LINE(' DESCRIPTION: ' || REC.DESCRIPTION);
   DBMS_OUTPUT.PUT_LINE(' COST: ' || REC.COST);
   DBMS_OUTPUT.PUT_LINE(' SELLER ID: ' || REC.SELLER_ID);
 END LOOP;
END;
/
```

Procedures

To purchase product(have to pass product id, quantity, customer id as argument)

```
CREATE OR REPLACE PROCEDURE PURCHASE_PRODUCT
(P_PRODUCT_ID NUMBER, P_QUANTITY NUMBER,V_CUSTOMER_ID
NUMBER)
IS
  V_PRODUCT_COST NUMBER(6);
  V_PRODUCT_QUANTITY NUMBER(10);
  V_TOTAL_COST NUMBER(10);
  V_PURCHASE_ID NUMBER(10);
  V_PURCHASE_DATE DATE := SYSDATE;
BEGIN
 SELECT COST, QUANTITY INTO V_PRODUCT_COST,
V_PRODUCT_QUANTITY FROM PRODUCT WHERE PRODUCT_ID =
P_PRODUCT_ID;
  V_TOTAL_COST := V_PRODUCT_COST * P_QUANTITY;
  INSERT INTO PURCHASE (PURCHASE_DATE, CUSTOMER_ID) VALUES
(V_PURCHASE_DATE, V_CUSTOMER_ID) RETURNING PURCHASE_ID
INTO V_PURCHASE_ID;
  INSERT INTO PURCHASE_ITEM (PURCHASE_ID, PRODUCT_ID,
SELLER_ID, QUANTITY) VALUES (V_PURCHASE_ID, P_PRODUCT_ID,
(SELECT SELLER_ID FROM PRODUCT WHERE PRODUCT_ID =
P_PRODUCT_ID), P_QUANTITY);
  DBMS_OUTPUT.PUT_LINE('PURCHASE SUCCESSFUL!');
  DBMS_OUTPUT.PUT_LINE('PRODUCT ID: ' || P_PRODUCT_ID);
  DBMS_OUTPUT.PUT_LINE('QUANTITY: ' || P_QUANTITY);
  DBMS_OUTPUT.PUT_LINE('TOTAL COST: $' || V_TOTAL_COST);
END;
/
```

Procedures

To display products less than given price

```
CREATE OR REPLACE PROCEDURE PRICE_LESSER(MINCOST NUMBER)
IS
CURSOR C1 IS (SELECT * FROM PRODUCT WHERE COST<=MINCOST);
BEGIN
 FOR I IN C1
 LOOP
   DBMS_OUTPUT.PUT_LINE('-----------------------------');
   DBMS_OUTPUT.PUT_LINE('PRODUCT ID: ' || I.PRODUCT_ID);
   DBMS_OUTPUT.PUT_LINE(' NAME: ' || I.NAME);
   DBMS_OUTPUT.PUT_LINE(' TYPE: ' || I.PROD_TYPE);
   DBMS_OUTPUT.PUT_LINE(' COLOUR: ' || I.COLOUR);
   DBMS_OUTPUT.PUT_LINE(' QUANTITY: ' || I.QUANTITY);
   DBMS_OUTPUT.PUT_LINE(' DESCRIPTION: ' || I.DESCRIPTION);
   DBMS_OUTPUT.PUT_LINE(' COST: ' || I.COST);
   DBMS_OUTPUT.PUT_LINE(' SELLER ID: ' || I.SELLER_ID);
   DBMS_OUTPUT.PUT_LINE('-----------------------------');
 END LOOP;
 END;
 /
```

Procedures

To get purchase record of a customer provided a customer id

```
CREATE OR REPLACE PROCEDURE GET_CUSTOMER_PURCHASES
(P_CUSTOMER_ID NUMBER) AS
 CURSOR C1 IS (SELECT *
  FROM PURCHASE P
   NATURAL JOIN PURCHASE_ITEM PI
  NATURAL JOIN PRODUCT P2  WHERE P.CUSTOMER_ID =
P_CUSTOMER_ID);
BEGIN
  FOR I IN C1
 LOOP
 DBMS_OUTPUT.PUT_LINE('PURCHASE ID: ' || I.PURCHASE_ID);
 DBMS_OUTPUT.PUT_LINE('PURCHASE DATE: ' || I.PURCHASE_DATE);
 DBMS_OUTPUT.PUT_LINE('PRODUCT: ' || I.NAME);
 DBMS_OUTPUT.PUT_LINE('QUANTITY: ' || I.QUANTITY);
 DBMS_OUTPUT.PUT_LINE('COST: ' || I.COST*I.QUANTITY);
 END LOOP;
END;
/
```

For seller to remove a product

```
CREATE OR REPLACE PROCEDURE REMOVE_PRODUCT(SID
NUMBER,PRODID NUMBER) AS
BEGIN
DELETE FROM PRODUCT WHERE PRODUCT_ID=PRODID AND
SELLER_ID=SID;
END;
/
```

Procedures

For seller to add Product

```
CREATE OR REPLACE PROCEDURE ADD_PRODUCT( V_NAME
VARCHAR,V_PROD_TYPE VARCHAR,V_COLOUR
VARCHAR,V_QUANTITY VARCHAR,V_DESCRIPTION
VARCHAR,V_COST NUMBER,V_SELLER_ID NUMBER)   AS
BEGIN
  INSERT INTO PRODUCT(NAME, PROD_TYPE, COLOUR, QUANTITY,
DESCRIPTION, COST, SELLER_ID)
  VALUES(V_NAME, V_PROD_TYPE, V_COLOUR, V_QUANTITY,
V_DESCRIPTION, V_COST, V_SELLER_ID);
  DBMS_OUTPUT.PUT_LINE('PRODUCT ADDED SUCCESSFULLY.');
END;
/
```

Triggers

If quantity purchased is more than available then purchase not allowed

```
CREATE OR REPLACE TRIGGER CHECK_PRODUCT_QUANTITY
BEFORE INSERT ON PURCHASE_ITEM
FOR EACH ROW
DECLARE
  V_PRODUCT_QUANTITY NUMBER(10);
BEGIN

  SELECT QUANTITY INTO V_PRODUCT_QUANTITY FROM PRODUCT
WHERE PRODUCT_ID = :NEW.PRODUCT_ID AND SELLER_ID =
:NEW.SELLER_ID;


  IF V_PRODUCT_QUANTITY < :NEW.QUANTITY THEN
    RAISE_APPLICATION_ERROR(-20001, 'NOT ENOUGH QUANTITY
AVAILABLE FOR PURCHASE');
  END IF;
END;
/
```

Triggers

Updates quantity in products table after purchase is made

```
CREATE OR REPLACE TRIGGER UPDATE_PRODUCT_QUANTITY
AFTER INSERT ON PURCHASE_ITEM
FOR EACH ROW
BEGIN
  UPDATE PRODUCT SET QUANTITY = QUANTITY - :NEW.QUANTITY
WHERE PRODUCT_ID = :NEW.PRODUCT_ID;
END;
/
```

Logs changes done to product table

```
CREATE OR REPLACE TRIGGER LOG_CHANGES_PRODUCT
BEFORE UPDATE OF QUANTITY,COST ON PRODUCT
FOR EACH ROW
BEGIN
INSERT INTO OLD_PRODUCT_TABLE
VALUES(CURRENT_TIMESTAMP,:OLD.PRODUCT_ID, :OLD.NAME,
:OLD.QUANTITY, :OLD.COST, :OLD.SELLER_ID);
END;
/
```

Triggers

To prevent the deletion of a seller who has products listed in the product table

```
CREATE OR REPLACE TRIGGER PREVENT_SELLER_DELETION
BEFORE DELETE ON SELLER
FOR EACH ROW
DECLARE
   PRODUCT_COUNT NUMBER(10);
BEGIN
   SELECT COUNT(*) INTO PRODUCT_COUNT FROM PRODUCT WHERE
SELLER_ID = :NEW.SELLER_ID;
   IF PRODUCT_COUNT > 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'CANNOT DELETE A SELLER
WITH LISTED PRODUCTS!');
   END IF;
END;
/
```

Automatically updates product_seller table when a product is added

```
CREATE OR REPLACE TRIGGER AD_TO_PRODSEL
AFTER INSERT ON PRODUCT
FOR EACH ROW
BEGIN
INSERT INTO PRODUCT_SELLER VALUES(:NEW.SELLER_ID,
:NEW.PRODUCT_ID);
END;
/
```

Our Team

G Gautam Datta

Registration No.: 210905394

Roll No.: 60

Kshiti Shetty

Registration No.: 210905137

Roll No.: 25

Bibliography

- https://123projectlab.com/er-diagram-for-online-shopping-system/
- https://fabric.inc/blog/shopping-cart-database-design/
- https://www.canva.com/design/DAFiZrfMKD8/EcSkDH1Xh7VPH2EKBd1i1g/edit