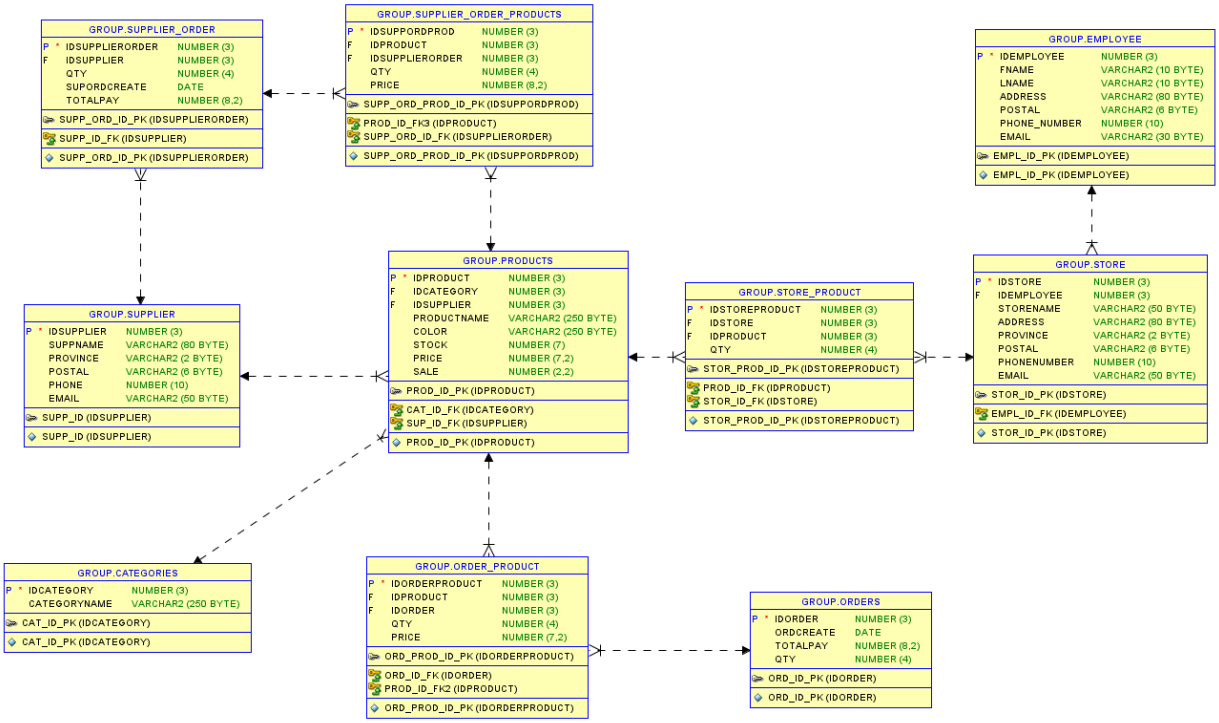
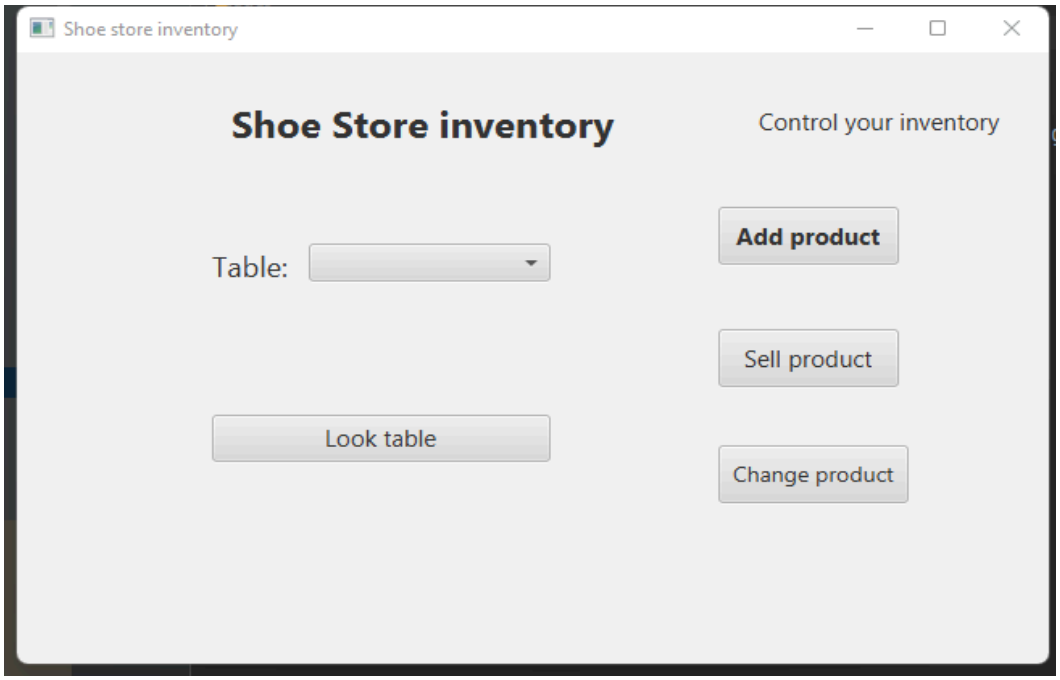
**Group members:**
Awas Jomail - 301210826
Juan Camilo Marmolejo - 301223791
Chihiro Hasegawa - 301229147
Yuichi Boki - 301216594

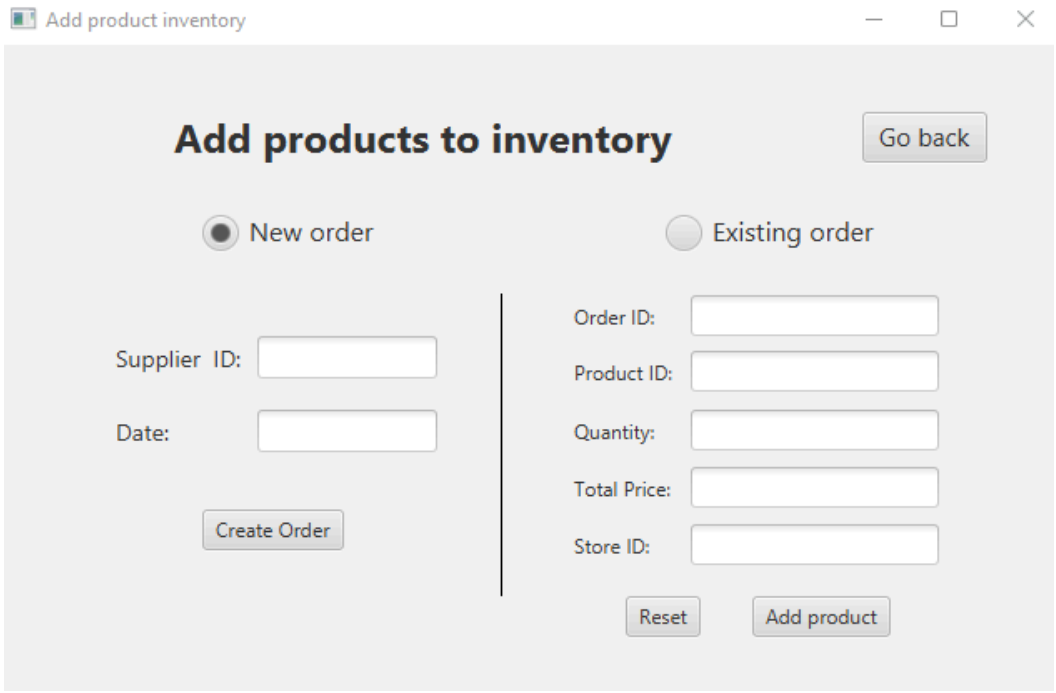
**Problem Domain**
The problem domain is inventory management including item stock updates. The business idea is to create a database system for a shoe store to update its inventory stock levels.

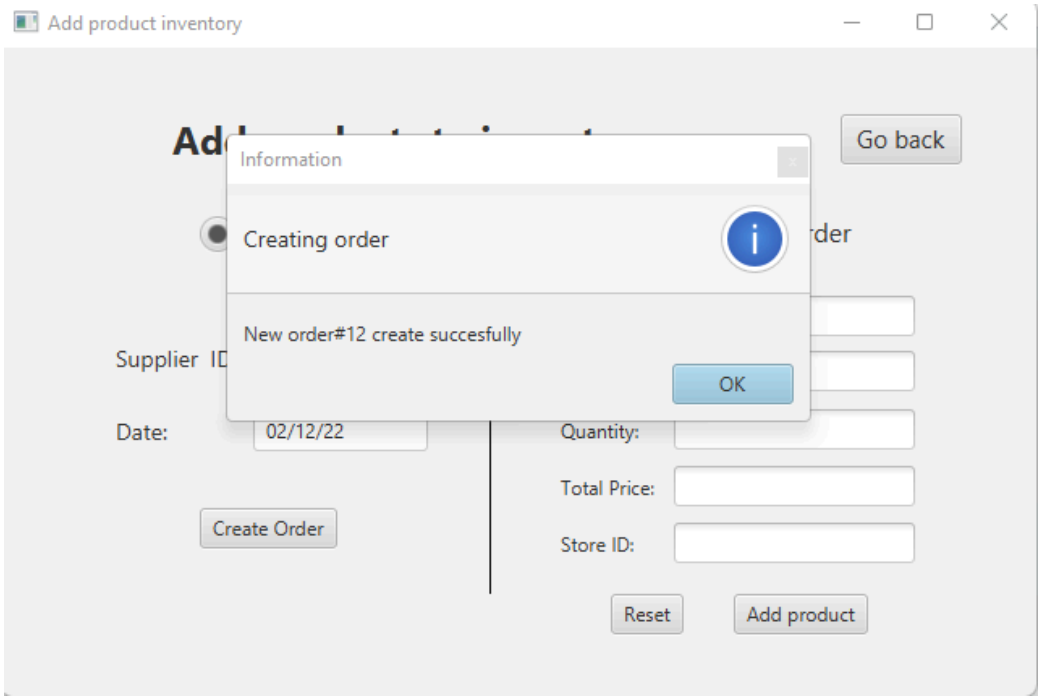
**ERD:**

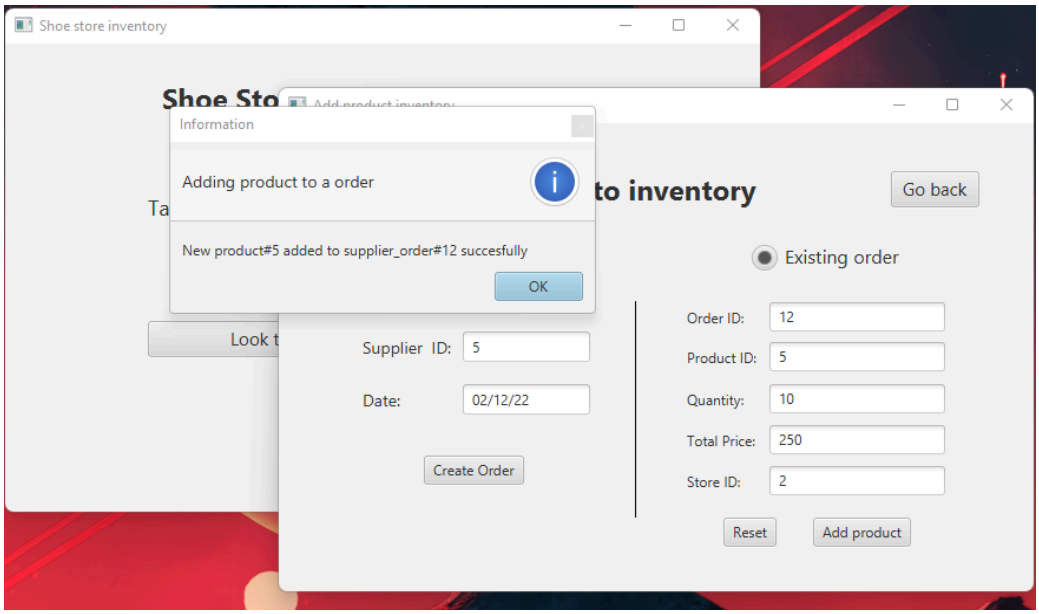


**Screenshots:**






**Front end: use java**

## SQL Code:

```
--------------------
-- TABLES--
--------------------
CREATE TABLE categories (
   idCategory NUMBER(3),
   categoryName VARCHAR2(250),
   CONSTRAINT cat_id_pk PRIMARY KEY(idCategory));

CREATE TABLE supplier (
   idSupplier NUMBER(3),
   suppName VARCHAR2(80),
   province VARCHAR2(2),
   postal VARCHAR2(6),
   phone NUMBER(10),
   email VARCHAR2(50),
   CONSTRAINT supp_id PRIMARY KEY(idSupplier) );

CREATE TABLE products (
   idProduct NUMBER(3),
   idCategory NUMBER(3),
   idSupplier NUMBER(3),
   productName VARCHAR2(250),
   color VARCHAR2(250),
   stock NUMBER(7),
   price NUMBER(7,2),
   sale NUMBER(2,2),
   CONSTRAINT prod_id_pk PRIMARY KEY(idProduct),
   CONSTRAINT cat_id_fk FOREIGN KEY(idCategory)
      REFERENCES categories(idCategory),
   CONSTRAINT sup_id_fk FOREIGN KEY(idSupplier)
      REFERENCES supplier(idSupplier) );

CREATE TABLE employee (
   idEmployee NUMBER(3) NOT NULL,
   fName VARCHAR2(10),
   lName VARCHAR2(10),
   address VARCHAR2(80),
   postal VARCHAR2(6),
   phone_number NUMBER(10),
```

```
    email VARCHAR2(30),
    CONSTRAINT empl_id_pk PRIMARY KEY(idEmployee) );

CREATE TABLE store (
    idStore NUMBER(3),
    idEmployee NUMBER(3),
    storeName VARCHAR2(50),
    address VARCHAR2(80),
    province VARCHAR2(2),
    postal VARCHAR2(6),
    phoneNumber NUMBER(10),
    email VARCHAR2(50),
    CONSTRAINT stor_id_pk PRIMARY KEY(idStore),
    CONSTRAINT empl_id_fk FOREIGN KEY(idEmployee)
        REFERENCES employee (idEmployee) );

CREATE TABLE store_product (
    idStoreProduct NUMBER(3),
    idStore NUMBER(3),
    idProduct NUMBER(3),
    qty NUMBER(4),
    CONSTRAINT stor_prod_id_pk PRIMARY KEY(idStoreProduct),
    CONSTRAINT stor_id_fk FOREIGN KEY(idStore)
        REFERENCES store (idStore),
    CONSTRAINT prod_id_fk FOREIGN KEY(idProduct)
        REFERENCES products (idProduct) );


CREATE TABLE orders (
    idOrder NUMBER(3),
    ordCreate DATE,
    totalPay NUMBER(8,2),
    qty NUMBER(4),
    CONSTRAINT ord_id_pk PRIMARY KEY(idOrder) );

CREATE TABLE order_product (
    idOrderProduct NUMBER(3),
    idProduct NUMBER(3),
    idOrder NUMBER(3),
    qty NUMBER(4),
    price NUMBER(7,2),
    CONSTRAINT ord_prod_id_pk PRIMARY KEY(idOrderProduct),
    CONSTRAINT prod_id_fk2 FOREIGN KEY(idProduct)
        REFERENCES products (idProduct),
```

```
    CONSTRAINT ord_id_fk FOREIGN KEY(idOrder)
       REFERENCES orders (idOrder) );



CREATE TABLE supplier_order (
    idSupplierOrder NUMBER(3),
    idSupplier NUMBER(3),
    qty NUMBER(4),
    supOrdCreate DATE,
    totalPay NUMBER(8,2),
    CONSTRAINT supp_ord_id_pk PRIMARY KEY(idSupplierOrder),
    CONSTRAINT supp_id_fk FOREIGN KEY (idSupplier)
       REFERENCES supplier (idSupplier) );

CREATE TABLE supplier_order_products (
    idSuppOrdProd NUMBER(3),
    idProduct NUMBER(3),
    idSupplierOrder NUMBER(3),
    qty NUMBER(4),
    price NUMBER(8,2),
    CONSTRAINT supp_ord_prod_id_pk PRIMARY KEY (idSuppOrdProd),
    CONSTRAINT prod_id_fk3 FOREIGN KEY (idProduct)
       REFERENCES products (idProduct),
    CONSTRAINT supp_ord_id_fk FOREIGN KEY (idSupplierOrder)
       REFERENCES supplier_order (idSupplierOrder) );



----------------------
-- SEQUENCES--
----------------------
CREATE SEQUENCE prod_id_seq;--
CREATE SEQUENCE ord_prod_id_seq;--
CREATE SEQUENCE cat_id_seq;--
CREATE SEQUENCE emp_id_seq;--
CREATE SEQUENCE supp_id_seq;--
CREATE SEQUENCE supp_ord_id_seq;--
CREATE SEQUENCE supp_ord_prod_id_seq;--
CREATE SEQUENCE stor_id_seq;
CREATE SEQUENCE stor_prod_id_seq;--
CREATE SEQUENCE ord_id_seq;--
```

```
---------------------
-- TRIGGERS, FUNCTIONS AND PROCEDURE--
---------------------
CREATE OR REPLACE TRIGGER trg_check_product_stock
  BEFORE INSERT ON order_product
  FOR EACH ROW
DECLARE
   var_stock NUMBER(4);
BEGIN
   select stock
   into var_stock
   from products
   where idproduct = :NEW.idproduct;

   IF var_stock = 0 THEN
      DBMS_OUTPUT.PUT_LINE('out of stock');
   ELSE
      DBMS_OUTPUT.PUT_LINE('in stock');
   END IF;
END;

-- test
INSERT INTO order_product
(idorderproduct, idproduct, idorder, qty, price)
VALUES
(ord_prod_id_seq.NEXTVAL, 5 , 1, 1, 130);


CREATE or replace TRIGGER update_stock_trg1
AFTER INSERT ON supplier_order_products
FOR EACH ROW
DECLARE
BEGIN

    UPDATE products
    SET stock = (select stock
                 from products
                 where idproduct = :NEW.idproduct ) + :NEW.qty
    where idproduct = :NEW.idproduct;
END update_stock_trg1;


--test
```

```
INSERT into supplier_order_products(IDSUPPORDPROD, IDPRODUCT, IDSUPPLIERORDER,
QTY, PRICE)
VALUES (supp_ord_prod_id_seq.nextval, 10, 1, 3, 250);


CREATE OR REPLACE PROCEDURE prod_ordered_products_counts
   (p_prodid IN NUMBER,
    p_order_num OUT NUMBER)
IS
   sorder_detail_id number(5) := 0;
   refcur_ordered_products SYS_REFCURSOR;
   idsuppordprod supplier_order_products.idsuppordprod%TYPE;
BEGIN
   OPEN refcur_ordered_products
   FOR
      select idsupplierorder
      from supplier_order_products
      where idproduct = p_prodid;
   LOOP
      FETCH refcur_ordered_products INTO idsuppordprod;
      EXIT WHEN refcur_ordered_products%NOTFOUND;
         IF idsuppordprod is not null
         THEN sorder_detail_id := sorder_detail_id + 1;
         END IF;
   END LOOP;
   p_order_num := sorder_detail_id;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error!');
END;

--- test
DECLARE
   p_order_num NUMBER;
   var_result NUMBER;
   var_idproduct supplier_order_products.idproduct%type := 5;
BEGIN
   prod_ordered_products_counts(var_idproduct, p_order_num);
   DBMS_OUTPUT.PUT_LINE(p_order_num);
END;


CREATE OR REPLACE
PROCEDURE update_sale_sp
```

```
(p_type IN VARCHAR2,
 idShoe IN products.idproduct%TYPE,
 p_sale IN products.sale%TYPE)
AS
CURSOR cur_product_prod is
(SELECT idproduct
 FROM products
 where idproduct = idShoe);
 CURSOR cur_product_cat is
(SELECT idproduct
 FROM products
 where idcategory = idShoe);
 CURSOR cur_product_supp is
(SELECT idproduct
 FROM products
 where idsupplier = idShoe);
 exception1 exception;
BEGIN
if (p_type = 'P' or p_type = 'p') then
DBMS_OUTPUT.PUT_LINE(p_type);
    FOR rec_product in cur_product_prod LOOP
    update products
    set sale = p_sale
    where idproduct = rec_product.idproduct;
    END LOOP;
elsif (p_type = 'C' or p_type = 'c') then
    DBMS_OUTPUT.PUT_LINE(p_type);
    FOR rec_product in cur_product_cat LOOP
    update products
    set sale = p_sale
    where idproduct = rec_product.idproduct;
    END LOOP;
ELSIF (p_type = 'B' or p_type = 'b') then
    DBMS_OUTPUT.PUT_LINE(p_type);
    FOR rec_product in cur_product_supp LOOP
    update products
    set sale = p_sale
    where idproduct = rec_product.idproduct;
    END LOOP;
else raise exception1;
end if;
exception
when exception1 then DBMS_OUTPUT.PUT_LINE('Error: please choose p for product, c for
category, and b for brand on the products you want to have sale on');
```

```
   END;
--test procedure 2
declare
begin
update_sale_sp('c',1,0.1);
end;

CREATE OR REPLACE FUNCTION new_products
(P_cat_id IN products.idcategory%TYPE,
 P_sup_id IN products.idsupplier%TYPE,
 P_prod_name IN products.productname%TYPE,
 P_prod_color IN products.color%TYPE,
 P_prod_stock IN products.stock%TYPE,
 P_prod_price IN products.price%TYPE,
 P_prod_sale IN products.sale%TYPE)
 RETURN VARCHAR2
IS

success_fun VARCHAR2(20):='INSERT IS SUCCESSFUL';
Seq_prod_id products.idproduct%TYPE;

BEGIN
 INSERT INTO products
   (idproduct, idcategory, idsupplier, productname, color, stock, price, sale)
 VALUES
         (prod_id_seq.NEXTVAL, P_cat_id, P_sup_id, P_prod_name, P_prod_color,
P_prod_stock, P_prod_price, P_prod_sale);
RETURN success_fun;
EXCEPTION
         WHEN VALUE_ERROR THEN
         DBMS_OUTPUT.PUT_LINE('SIZE ERROR');
         WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE('NO DATA');--for idcategory
         WHEN OTHERS THEN
         DBMS_OUTPUT.PUT_LINE('SOMETHING IS WRONG');

END new_products;


--test--
begin
DBMS_OUTPUT.PUT_LINE(new_products(5, 9, 'BLUE NOVA', 'BLUE', 10, 160, 0));

END;
```

```
CREATE OR REPLACE
FUNCTION is_expensive_sf
   (idProd IN products.idproduct%TYPE)
   RETURN VARCHAR2
   AS
   my_price products.price%type;
   BEGIN
   SELECT price
   INTO my_price
   FROM products
   WHERE idproduct = idProd;
      IF my_price <= 100 THEN
      RETURN 'The shoe is '||my_price||', which is affordable';
      elsif my_price <= 200 AND my_price > 100 THEN
      RETURN 'The shoe is '||my_price||', which is a bit expensive';
      else RETURN 'The shoe is '||my_price||', which is expensive';
      end if;
   END;

--test
DECLARE
BEGIN
DBMS_OUTPUT.PUT_LINE(is_expensive_sf(6));
END;

CREATE OR REPLACE FUNCTION new_supplier_order
(S_supplier_ID IN supplier_order.idsupplier%TYPE,
 S_supplier_date IN supplier_order.supordcreate%TYPE)
 RETURN VARCHAR2
IS
S_supplierorder_ID supplier_order.idsupplierorder%TYPE:=supp_ord_id_seq.NEXTVAL ;
S_supplier_qty supplier_order.qty%TYPE := 0;
S_supplier_totalpay supplier_order.totalpay%TYPE := 0;
lv_message VARCHAR2(35);
BEGIN

 INSERT INTO supplier_order
 VALUES
          (S_supplierorder_ID, S_supplier_ID, S_supplier_qty, S_supplier_date,
S_supplier_totalpay);
 lv_message:= 'New order#' || S_supplierorder_ID || ' create succesfully';
RETURN lv_message;
```

```
EXCEPTION
          WHEN VALUE_ERROR THEN
       RETURN 'SIZE ERROR';
          WHEN NO_DATA_FOUND THEN
     RETURN 'NO DATA';--for idcategory
          WHEN OTHERS THEN
     RETURN 'SOMETHING IS WRONG';
END new_supplier_order;


--test--
BEGIN
DBMS_OUTPUT.PUT_LINE(new_supplier_order(5,SYSDATE));
END;

CREATE OR REPLACE FUNCTION new_supplier_order_products
(S_supplierorder_ID IN supplier_order_products.idsupplierorder%TYPE,
 S_supplierproducts_productID IN supplier_order_products.idproduct%TYPE,
 S_supplierproducts_qty IN supplier_order_products.qty%TYPE,
 S_supplierproducts_price IN supplier_order_products.price%TYPE)
 RETURN VARCHAR2
IS
S_supplierorderproducts_ID
supplier_order_products.idsuppordprod%TYPE:=supp_ord_prod_id_seq.NEXTVAL ;
lv_message VARCHAR2(100);
BEGIN

 INSERT INTO supplier_order_products
 VALUES
          (S_supplierorderproducts_ID, S_supplierproducts_productID, S_supplierorder_ID,
S_supplierproducts_qty, S_supplierproducts_price);
 lv_message:= 'New product#' || S_supplierproducts_productID ||' added to supplier_order#' ||
S_supplierorder_ID || ' succesfully';
RETURN lv_message;
EXCEPTION
          WHEN VALUE_ERROR THEN
       RETURN 'ERROR ADDING PRODUCT: SIZE ERROR';
          WHEN NO_DATA_FOUND THEN
     RETURN 'ERROR ADDING PRODUCT: NO DATA';
          WHEN OTHERS THEN
     RETURN 'ERROR ADDING PRODUCT: SOMETHING IS WRONG';
END new_supplier_order_products;
```

```
-- package specification
create or replace PACKAGE abc_shoes_pkg IS
    -- global variable
    msg VARCHAR(100);

    -- procedure#1
    PROCEDURE prod_ordered_products_counts
    (p_prodid IN NUMBER,
     p_order_num OUT NUMBER);

    -- procedure#2
    PROCEDURE update_sale_sp
    (p_type IN VARCHAR2,
     idShoe IN products.idproduct%TYPE,
     p_sale IN products.sale%TYPE);

    -- function#1
    FUNCTION new_products
    (P_cat_id IN products.idcategory%TYPE,
     P_sup_id IN products.idsupplier%TYPE,
     P_prod_name IN products.productname%TYPE,
     P_prod_color IN products.color%TYPE,
     P_prod_stock IN products.stock%TYPE,
     P_prod_price IN products.price%TYPE,
     P_prod_sale IN products.sale%TYPE)
     RETURN VARCHAR2;

    -- function#2
    FUNCTION is_expensive_sf
    (idProd IN products.idproduct%TYPE)
    RETURN VARCHAR2;
END;

-- package body
create or replace PACKAGE BODY abc_shoes_pkg IS

    -- private variable
    pro_fun VARCHAR(15);

    -- procedure#1
    PROCEDURE prod_ordered_products_counts
        (p_prodid IN NUMBER,
         p_order_num OUT NUMBER)
    IS
```

```
    sorder_detail_id number(5) := 0;
    refcur_ordered_products SYS_REFCURSOR;
    idsuppordprod supplier_order_products.idsuppordprod%TYPE;
BEGIN
    OPEN refcur_ordered_products
    FOR
        select idsupplierorder
        from supplier_order_products
        where idproduct = p_prodid;
    LOOP
        FETCH refcur_ordered_products INTO idsuppordprod;
        EXIT WHEN refcur_ordered_products%NOTFOUND;
            IF idsuppordprod is not null
            THEN sorder_detail_id := sorder_detail_id + 1;
            END IF;
    END LOOP;
    p_order_num := sorder_detail_id;

    -- show global variable and private variable
    pro_fun := 'Procedure#1';
    msg := pro_fun || ' done!';
    DBMS_OUTPUT.PUT_LINE(msg);

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error!');
END prod_ordered_products_counts;

-- procedure#2
PROCEDURE update_sale_sp
(p_type IN VARCHAR2,
 idShoe IN products.idproduct%TYPE,
 p_sale IN products.sale%TYPE)
AS
CURSOR cur_product_prod is
(SELECT idproduct
 FROM products
 where idproduct = idShoe);
 CURSOR cur_product_cat is
(SELECT idproduct
 FROM products
 where idcategory = idShoe);
 CURSOR cur_product_supp is
(SELECT idproduct
```

```
 FROM products
 where idsupplier = idShoe);
 exception1 exception;
BEGIN
if (p_type = 'P' or p_type = 'p') then
DBMS_OUTPUT.PUT_LINE(p_type);
   FOR rec_product in cur_product_prod LOOP
   update products
   set sale = p_sale
   where idproduct = rec_product.idproduct;
   END LOOP;
elsif (p_type = 'C' or p_type = 'c') then
   DBMS_OUTPUT.PUT_LINE(p_type);
   FOR rec_product in cur_product_cat LOOP
   update products
   set sale = p_sale
   where idproduct = rec_product.idproduct;
   END LOOP;
ELSIF (p_type = 'B' or p_type = 'b') then
   DBMS_OUTPUT.PUT_LINE(p_type);
   FOR rec_product in cur_product_supp LOOP
   update products
   set sale = p_sale
   where idproduct = rec_product.idproduct;
   END LOOP;
else raise exception1;
end if;

   -- show global variable and private variable
   pro_fun := 'Procedure#2';
   msg := pro_fun || ' done!';
   DBMS_OUTPUT.PUT_LINE(msg);

exception
when exception1 then DBMS_OUTPUT.PUT_LINE('Error: please choose p for product, c for
category, and b for brand on the products you want to have sale on');
END update_sale_sp;

--function#1
FUNCTION new_products
(P_cat_id IN products.idcategory%TYPE,
 P_sup_id IN products.idsupplier%TYPE,
 P_prod_name IN products.productname%TYPE,
 P_prod_color IN products.color%TYPE,
```

```
    P_prod_stock IN products.stock%TYPE,
    P_prod_price IN products.price%TYPE,
    P_prod_sale IN products.sale%TYPE)
    RETURN VARCHAR2
   IS

   success_fun VARCHAR2(20):='INSERT IS SUCCESSFUL';
   Seq_prod_id products.idproduct%TYPE;

   BEGIN
    INSERT INTO products
      (idproduct, idcategory, idsupplier, productname, color, stock, price, sale)
    VALUES
      (prod_id_seq.NEXTVAL, P_cat_id, P_sup_id, P_prod_name, P_prod_color, P_prod_stock,
P_prod_price, P_prod_sale);

   -- show global variable and private variable
   pro_fun := 'Function#1';
   msg := pro_fun || ' done!';
   DBMS_OUTPUT.PUT_LINE(msg);

   RETURN success_fun;
   EXCEPTION
      WHEN VALUE_ERROR THEN
      DBMS_OUTPUT.PUT_LINE('SIZE ERROR');
      WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('NO DATA');--for idcategory
      WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('SOMETHING IS WRONG');

   END new_products;

   -- function#2
   FUNCTION is_expensive_sf
   (idProd IN products.idproduct%TYPE)
   RETURN VARCHAR2
   AS
   my_price products.price%type;
   BEGIN
   SELECT price
   INTO my_price
   FROM products
   WHERE idproduct = idProd;
```

```
    -- show global variable and private variable
    pro_fun := 'Function#2';
    msg := pro_fun || ' done!';
    DBMS_OUTPUT.PUT_LINE(msg);

        IF my_price <= 100 THEN
        RETURN 'The shoe is '||my_price||', which is affordable';
        elsif my_price <= 200 AND my_price > 100 THEN
        RETURN 'The shoe is '||my_price||', which is a bit expensive';
        else RETURN 'The shoe is '||my_price||', which is expensive';
        end if;
    END is_expensive_sf;

END;


--Package Testing
declare
lv_order_num number;
begin
abc_shoes_pkg.update_sale_sp('p',1,0.1);
abc_shoes_pkg.prod_ordered_products_counts(1,lv_order_num);
DBMS_OUTPUT.PUT_LINE(abc_shoes_pkg.new_products(1, 1, 'awas brand', 'black', 1, 500,
null));
DBMS_OUTPUT.pUT_lINE(abc_shoes_pkg.is_expensive_sf(1));
end;
```