- **Procedure**

```
CREATE OR REPLACE PROCEDURE sumprice AS
BEGIN
 FOR prodt IN (
SELECt user_id, sum(total_price) as pr_sum
FROM wishlist
GROUP BY user_id) LOOP
DBMS_OUTPUT.PUT_LINE('Users id: ' || prodt.user_id || ', Total price:
' ||prodt.pr_sum);
 END LOOP;
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Unfortunately, this error occurred: ' ||
SQLERRM);
END;

Begin
Sumprice;
End;
```

- **Function**

```
CREATE OR REPLACE FUNCTION num_of_orders_to_shipment
(c_orders IN
SYS_REFCURSOR)
RETURN NUMBER IS
    c_order_id orders.order_id%TYPE;
    c_order_status orders.order_status%TYPE;
    cnt number:=0;
BEGIN
    LOOP
    FETCH c_orders INTO c_order_id, c_order_status;
    EXIT WHEN c_orders%NOTFOUND;
```

```
        IF c_order_status = 3 or c_order_status = 4 THEN
                cnt:=cnt+1;
        END IF;
        END LOOP;
        CLOSE c_orders;
        RETURN cnt;
END;
/



DECLARE
        cnt_of_orders_to_shipment number:=0;
        c_orders SYS_REFCURSOR;
BEGIN
        OPEN c_orders FOR
        SELECT order_id, order_status FROM orders;
        cnt_of_orders_to_shipment := num_of_orders_to_shipment(c_orders);
        dbms_output.put_line('Number of orders that can go to shipment is: ' ||
cnt_of_orders_to_shipment);
 END;
 /
```

- **User-Defined Exception**

```
    DECLARE
PROD_ID PRODUCTS.PROD_ID%TYPE;
PROD_NAME PRODUCTS.PROD_NAME%TYPE;
COMPANY_ID PRODUCTS.COMPANY_ID%TYPE;
PROD_TYPE PRODUCT_CATEGORY.PROD_TYPE%TYPE;
invalid_name EXCEPTION;
BEGIN
PROD_ID := :PROD_ID;
PROD_NAME := :PROD_NAME;
COMPANY_ID := :COMPANY_ID;
PROD_TYPE := :PROD_TYPE;

IF LENGTH(PROD_NAME) < 5 THEN
RAISE invalid_name;
END IF;
INSERT INTO PRODUCTS (PROD_ID, PROD_NAME, COMPANY_ID)
```

```
VALUES (PROD_ID, PROD_NAME, COMPANY_ID);
INSERT INTO PRODUCT_CATEGORY (PROD_ID, PROD_TYPE)
VALUES (PROD_ID, PROD_TYPE);


EXCEPTION
WHEN invalid_name THEN
dbms_output.put_line('Invalid product name');
END;
```

- **SQL%Rowcount**

```
CREATE OR REPLACE PROCEDURE upd_price
IS
  s_cnt NUMBER;
BEGIN
    update PRODUCTS_INFO
    set price = price * 1.01;
    s_cnt := SQL%ROWCOUNT;
    IF s_cnt = 0 then
        DBMS_OUTPUT.PUT_LINE('No rows updted');
    ELSE
        DBMS_OUTPUT.PUT_LINE(s_cnt || 'rows updated');
    END IF;
END;

BEGIN
  upd_price;
END;
```

- **Trigger**

```
CREATE OR REPLACE TRIGGER current_number_of_rows
BEFORE INSERT ON  products_info
DECLARE
  number_of_rows NUMBER;
```

```
BEGIN
  SELECT COUNT(*) INTO number_of_rows FROM  products_info ;
  DBMS_OUTPUT.PUT_LINE('Current number of rows in the table: ' ||
number_of_rows);
END;
/
```