\*\*\*\*item→ property palette → conceal data = “yes”

Begin

SET\_ITEM\_PROPERTY(:MEMBER\_LOGIN\_C.MEM\_PASSWORD\_TXB, CONCEAL\_DATA, PROPERTY\_TRUE);

END;

DECLARE

POINTS NUMBER(20);

INI\_POINTS NUMBER(20);

TOTAL NUMBER (20);

RETURN\_VALUE NUMBER(20);

username\_count number(3);

STARS NUMBER(20);

STARS\_REDEEMED NUMBER(20);

BEGIN

SELECT COUNT(\*) INTO username\_count from MEMBER

where MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

IF username\_count = 0 THEN

set\_item\_property('REDEEM\_C.PTS\_WANT\_TO\_REDEEM',Update\_allowed,Property\_false);

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Walk-in customer can not redeem points.');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

ELSIF LENGTH(TRIM(TRANSLATE(:REDEEM\_C.PTS\_WANT\_TO\_REDEEM, 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ', ' '))) is NULL THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Redeemed points must be number');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

ELSIF LENGTH(:REDEEM\_C.PTS\_WANT\_TO\_REDEEM) <> LENGTH(TRIM(TRANSLATE(:REDEEM\_C.PTS\_WANT\_TO\_REDEEM, 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ', ' '))) THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Redeemed points must be number');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

ELSIF FLOOR(:REDEEM\_C.PTS\_WANT\_TO\_REDEEM) <> CEIL(:REDEEM\_C.PTS\_WANT\_TO\_REDEEM) THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Redeemed points must be integer');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

ELSIF :REDEEM\_C.PTS\_WANT\_TO\_REDEEM <0 THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Redeemed points cannot be negative number');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

elsif :REDEEM\_C.PTS\_WANT\_TO\_REDEEM >:REDEEM\_C.PTS\_EARNED THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'You do not have enough points.');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

ELSIF :REDEEM\_C.PTS\_WANT\_TO\_REDEEM > :SC\_C.TOTAL THEN

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'Redeemed points exceed the total price.');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

END IF;

SELECT MEM\_STAR INTO STARS FROM MEMBER WHERE MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

INI\_POINTS := :REDEEM\_C.PTS\_EARNED;

POINTS := :REDEEM\_C.PTS\_WANT\_TO\_REDEEM;

TOTAL := :SC\_C.TOTAL;

TOTAL := TOTAL - POINTS;

:SC\_C.TOTAL := TOTAL;

:REDEEM\_C.PTS\_EARNED := INI\_POINTS - POINTS;

STARS\_REDEEMED := STARS - :REDEEM\_C.PTS\_EARNED;

SET\_ALERT\_PROPERTY('PTS\_ALERT', ALERT\_MESSAGE\_TEXT, 'You have redeemed ' || STARS\_REDEEMED || ' points totally.');

RETURN\_VALUE := SHOW\_ALERT('PTS\_ALERT');

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := STARS - :REDEEM\_C.PTS\_EARNED;

END;

REDEEM POINT CLEAR

DECLARE

username\_count number(3);

TOTALAMOUNT NUMBER(20);

BEGIN

SELECT COUNT(\*) INTO username\_count from MEMBER

where MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

if username\_count = 1 then

SELECT MEM\_STAR INTO :REDEEM\_C.PTS\_EARNED FROM MEMBER

WHERE MEM\_UNAME = :MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

else

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

RAISE FORM\_TRIGGER\_FAILURE;

END IF;

GO\_BLOCK('SCLIST\_C');

FIRST\_RECORD;

TOTALAMOUNT := 0;

LOOP

GO\_BLOCK('SC\_C');

TOTALAMOUNT := TOTALAMOUNT + :SCLIST\_C.PRODUCT\_PRICE;

:SC\_C.TOTAL := TOTALAMOUNT;

GO\_BLOCK('SCLIST\_C');

NEXT\_RECORD;

EXIT WHEN :SCLIST\_C.PRODUCT\_PRICE IS NULL;

END LOOP;

:REDEEM\_C.PTS\_WANT\_TO\_REDEEM := 0;

END;

CHECK OUT

--UPDATE PTS

DECLARE

PTS NUMBER(20);

username\_count NUMBER(3);

STARS NUMBER(20);

BEGIN

PTS := :FIRMORDER\_C.TOTALPT;

SELECT COUNT(\*) INTO username\_count from MEMBER

WHERE MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

IF username\_count = 1 then

SELECT MEM\_STAR INTO STARS FROM MEMBER WHERE MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

IF PTS <> STARS then

UPDATE MEMBER

SET MEM\_STAR = PTS

WHERE MEM\_UNAME = :MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

forms\_ddl ('Commit');

ELSE NULL;

END IF;

IF :FIRMORDER\_C.TOTAL >= 20 THEN--AND username\_count = 1

UPDATE MEMBER

SET MEM\_STAR = MEM\_STAR + TRUNC(:FIRMORDER\_C.TOTAL / 20)

WHERE MEM\_UNAME = :MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

forms\_ddl ('Commit');

ELSE NULL;

END IF;

END IF;

END;

-- CREATE NEW ORDER ID

DECLARE

CURRENT\_ORDER\_ID NUMBER(20);

NEW\_ORDER\_ID NUMBER(20);

MEMID NUMBER(20);

username\_count number(20);

BEGIN

SELECT COUNT(\*) INTO username\_count from MEMBER

where MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

if username\_count=1 then

SELECT MEM\_ID INTO MEMID FROM MEMBER WHERE MEM\_UNAME = :MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

SELECT MAX(ORDERS\_ID) INTO CURRENT\_ORDER\_ID FROM ORDERS;

NEW\_ORDER\_ID:=CURRENT\_ORDER\_ID+1;

INSERT INTO ORDERS (ORDERS\_ID,ORDERS\_STATUS,ORDERS\_DISAMOUNT,MEM\_ID,ORDERS\_AMOUNT) VALUES (NEW\_ORDER\_ID,'PENDING',:FIRMORDER\_C.TOTAL,MEMID,:FIRMORDER\_C.TOTAL+:FIRMORDER\_C.REDEEMED);

forms\_ddl ('Commit');

elsif username\_count=0 then

SELECT MAX(ORDERS\_ID) INTO CURRENT\_ORDER\_ID FROM ORDERS;

NEW\_ORDER\_ID:=CURRENT\_ORDER\_ID+1;

INSERT INTO ORDERS (ORDERS\_ID,ORDERS\_STATUS,ORDERS\_DISAMOUNT,ORDERS\_AMOUNT) VALUES (NEW\_ORDER\_ID,'PENDING',:FIRMORDER\_C.TOTAL,:FIRMORDER\_C.TOTAL+:FIRMORDER\_C.REDEEMED);

forms\_ddl ('Commit');

end if;

END;

DECLARE

Current\_DETAIL\_ID NUMBER (20);

New\_DETAIL\_ID NUMBER(20);

PROD\_ID NUMBER(20);

ORD\_ID NUMBER(20);

username\_count number(20);

I NUMBER(3);

BEGIN

SELECT COUNT(\*) INTO username\_count from MEMBER

where MEM\_UNAME=:MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

FIRST\_RECORD;

LOOP

GO\_BLOCK('CONORDER\_C');

--Generate DETAIL\_ID

SELECT MAX (DETAIL\_ID) INTO Current\_DETAIL\_ID FROM DETAIL;

New\_DETAIL\_ID:=Current\_DETAIL\_ID+1;

SELECT PRODUCT\_ID INTO PROD\_ID FROM PRODUCT

WHERE PRODUCT\_NAME = Lower(:CONORDER\_C.PRODUCT\_NAME);

SELECT MAX(ORDERS\_ID) INTO ORD\_ID FROM ORDERS;

INSERT INTO DETAIL (DETAIL\_ID,DETAIL\_TEMPERATURE,DETAIL\_COLDNESS,DETAIL\_SWEETNESS,

DETAIL\_ADDPARTS,DETAIL\_QUANTITY,DETAIL\_STATUS,PRODUCT\_ID,ORDERS\_ID)

VALUES (New\_DETAIL\_ID,:CONORDER\_C.DETAIL\_TEMPERATURE,:CONORDER\_C.DETAIL\_COLDNESS,

:CONORDER\_C.DETAIL\_SWEETNESS,:CONORDER\_C.DETAIL\_ADDPARTS,

:CONORDER\_C.DETAIL\_QUANTITY,'PENDING',PROD\_ID,ORD\_ID);

forms\_ddl('Commit');

NEXT\_RECORD;

Exit when :CONORDER\_C.PRODUCT\_NAME IS NULL;

END LOOP;

--COPY TRASACTION DETAILS TO NEXT PAGE

IF username\_count = 1 THEN

GO\_BLOCK('TRANS\_COMPLETE\_MEM\_C');

:TRANS\_COMPLETE\_MEM\_C.COMPLETE\_MEM\_UNAME := :MEMBER\_LOGIN\_C.MEM\_USERNAME\_TXB;

:TRANS\_COMPLETE\_MEM\_C.COMPLETE\_MEM\_AMOUNT := '$'||:FIRMORDER\_C.TOTAL;

GO\_BLOCK('CONORDER\_C');

FIRST\_RECORD;

LOOP

:COMPLETE\_MEM\_ITEMS\_C.PRODUCT\_NAME:=:CONORDER\_C.PRODUCT\_NAME;

:COMPLETE\_MEM\_ITEMS\_C.PRODUCT\_PRICE:=:CONORDER\_C.PRODUCT\_PRICE;

:COMPLETE\_MEM\_ITEMS\_C.DETAIL\_SWEETNESS := :CONORDER\_C.DETAIL\_SWEETNESS;

:COMPLETE\_MEM\_ITEMS\_C.DETAIL\_TEMPERATURE:= :CONORDER\_C.DETAIL\_TEMPERATURE;

:COMPLETE\_MEM\_ITEMS\_C.DETAIL\_COLDNESS:= :CONORDER\_C.DETAIL\_COLDNESS;

:COMPLETE\_MEM\_ITEMS\_C.DETAIL\_ADDPARTS := :CONORDER\_C.DETAIL\_ADDPARTS;

:COMPLETE\_MEM\_ITEMS\_C.DETAIL\_QUANTITY := :CONORDER\_C.DETAIL\_QUANTITY;

GO\_BLOCK('COMPLETE\_MEM\_ITEMS\_C');

NEXT\_RECORD;

GO\_BLOCK('CONORDER\_C');

NEXT\_RECORD;

EXIT WHEN :CONORDER\_C.PRODUCT\_NAME IS NULL;

END LOOP;

GO\_BLOCK('TRANS\_COMPLETE\_MEM\_C');

END IF;

IF username\_count = 0 THEN

GO\_BLOCK('TRANS\_COMPLETE\_C');

:TRANS\_COMPLETE\_C.COMPLETE\_NONMEM\_AMOUNT := '$'||:FIRMORDER\_C.TOTAL;

GO\_BLOCK('CONORDER\_C');

FIRST\_RECORD;

LOOP

:COMPLETE\_ITEMS\_C.PRODUCT\_NAME:=:CONORDER\_C.PRODUCT\_NAME;

:COMPLETE\_ITEMS\_C.PRODUCT\_PRICE := :CONORDER\_C.PRODUCT\_PRICE;

:COMPLETE\_ITEMS\_C.DETAIL\_SWEETNESS := :CONORDER\_C.DETAIL\_SWEETNESS;

:COMPLETE\_ITEMS\_C.DETAIL\_TEMPERATURE:= :CONORDER\_C.DETAIL\_TEMPERATURE;

:COMPLETE\_ITEMS\_C.DETAIL\_COLDNESS:= :CONORDER\_C.DETAIL\_COLDNESS;

:COMPLETE\_ITEMS\_C.DETAIL\_ADDPARTS := :CONORDER\_C.DETAIL\_ADDPARTS;

:COMPLETE\_ITEMS\_C.DETAIL\_QUANTITY := :CONORDER\_C.DETAIL\_QUANTITY;

GO\_BLOCK('COMPLETE\_ITEMS\_C');

NEXT\_RECORD;

GO\_BLOCK('CONORDER\_C');

NEXT\_RECORD;

EXIT WHEN :CONORDER\_C.PRODUCT\_NAME IS NULL;

END LOOP;

GO\_BLOCK('TRANS\_COMPLETE\_C');

END IF;

END;