**SOLUTION**

**REQUIREMENT 1**

CREATE OR REPLACE PROCEDURE get\_pending\_order(p\_ostatus IN TBLorderMaster.ostatus%TYPE)

AS

CURSOR c\_get\_status IS

SELECT \* FROM TBLorderMaster

WHERE ostatus = p\_ostatus;

v\_order TBLorderMaster%ROWTYPE;

BEGIN

OPEN c\_get\_status;

LOOP

FETCH c\_get\_status INTO v\_order;

EXIT WHEN c\_get\_status%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('We have'||' '||v\_order.orderid||' '||'in'||' '||v\_order.ostatus||' '||'status');

END LOOP;

END;

**TO EXECUTE**

BEGIN

get\_pending\_order('P');

END;

**----------------------------------------------------------------------------------------------------------------------------------------**

**REQUIREMENT 2**

CREATE OR REPLACE PROCEDURE get\_order\_quantity(v\_dummy in NUMBER , p\_order\_id in TBLorderMaster.orderid%TYPE)

IS

TYPE ref\_cur is REF CURSOR;

refvar1 ref\_cur;

v\_date DATE;

v\_order VARCHAR2(10);

v\_item VARCHAR2(10);

BEGIN

IF v\_dummy = 10 THEN

OPEN refvar1 FOR

SELECT odate FROM TBLorderMaster

WHERE orderid = p\_order\_id;

FETCH refvar1 INTO v\_date;

DBMS\_OUTPUT.PUT\_LINE('Order date is'||' '|| v\_date);

CLOSE refvar1;

ELSE

OPEN refvar1 FOR

SELECT qty\_ordered, itemid FROM TBLOrderDtl

WHERE orderid = p\_order\_id;

LOOP

FETCH refvar1 INTO v\_order,v\_item;

EXIT WHEN refvar1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Quantity Ordered'||' '||'for'||' '||v\_item||' '||'is'||' '||v\_order);

END LOOP;

CLOSE refvar1;

END IF;

END;

**TO EXECUTE**

BEGIN

get\_order\_quantity(10,'o001');

END;

------------------------------------------------------------------------------------------------------------------------------------------

**REQUIREMENT 3**

CREATE OR REPLACE PROCEDURE check\_delivery(p\_order\_id IN TBLOrderDtl.orderid%TYPE)

AS

CURSOR c\_check\_delivery IS

SELECT \*

FROM TBLOrderDtl

WHERE qty\_delivered < qty\_ordered

AND orderid = p\_order\_id;

v\_check c\_check\_delivery%ROWTYPE;

v\_balanceOrders NUMBER;

BEGIN

OPEN c\_check\_delivery;

LOOP

FETCH c\_check\_delivery INTO v\_check;

EXIT WHEN c\_check\_delivery%NOTFOUND;

v\_balanceOrders := v\_check.qty\_ordered - v\_check.qty\_delivered;

DBMS\_OUTPUT.PUT\_LINE('The item code'||' '||v\_check.itemid||' '||'with order ID'||' '||v\_check.orderid||' '||'need to deliver'||' '||v\_balanceOrders||' '||'more orders');

END LOOP;

END;

**TO EXECUTE**

BEGIN

check\_delivery('o001');

END;

--------------------------------------------------------------------------------------------------------------------------------------

**REQUIREMENT 4**

CREATE OR REPLACE FUNCTION func\_getmax(f\_item\_id IN TBLitem.itemid%TYPE)

RETURN VARCHAR2 AS

v\_get\_max TBLitem%ROWTYPE;

v\_max\_level VARCHAR2(100);

BEGIN

SELECT \* INTO v\_get\_max

FROM TBLitem

WHERE itemid = f\_item\_id;

v\_max\_level := v\_get\_max.p\_category||' '||'with'||' '||'item id'||' '||v\_get\_max.itemid||' '||'maximum level is'||' '||v\_get\_max.max\_level;

RETURN v\_max\_level;

END;

**TO EXECUTE**

DECLARE

a VARCHAR2(100);

BEGIN

a := func\_getmax('i101');

DBMS\_OUTPUT.PUT\_LINE(a);

END;

---------------------------------------------------------------------------------------------------------------------------------------

**REQUIREMENT 5**

CREATE OR REPLACE TRIGGER trigger\_order

BEFORE INSERT OR UPDATE ON TBLOrderDtl

FOR EACH ROW

BEGIN

IF INSERTING THEN

IF :NEW.qty\_delivered > :NEW.qty\_ordered THEN

RAISE\_APPLICATION\_ERROR (-20001,'Quantity delivered should not be greater than Quantity Ordered');

END IF;

ELSIF UPDATING THEN

IF :NEW.qty\_delivered > :OLD.qty\_ordered THEN

RAISE\_APPLICATION\_ERROR (-20001,'Quantity delivered should not be greater than Quantity Ordered');

END IF;

END IF;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

END;

**TO EXECUTE**

INSERT INTO TBLOrderDtl(OrderID,ItemID,Qty\_Ordered,Qty\_Delivered)

VALUES('o001','i103',100,145);

UPDATE TBLOrderDtl SET Qty\_Delivered = 145

WHERE orderid ='o001'

AND itemid = 'i103'

------------------------------------------------------------------------------------------------------------------------------------------

**REQUIREMENT 6**

CREATE OR REPLACE PROCEDURE proc\_invalid\_item (p\_item\_id IN TBLitem.Itemid%TYPE)

AS

ex\_user\_invalid\_id EXCEPTION;

rec\_item TBLitem%ROWTYPE;

BEGIN

IF p\_item\_id <= 0 THEN

RAISE ex\_user\_invalid\_id;

ELSE

SELECT \* INTO rec\_item

FROM TBLitem

WHERE itemid = p\_item\_id;

DBMS\_OUTPUT.PUT\_LINE ('Item Description: '|| rec\_item.itemdesc);

DBMS\_OUTPUT.PUT\_LINE ('Product category: ' || rec\_item.p\_category);

END IF;

EXCEPTION

WHEN ex\_user\_invalid\_id THEN

dbms\_output.put\_line('Item ID must not be Zero!');

WHEN no\_data\_found THEN

dbms\_output.put\_line('No such Item!');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

**TO EXECUTE**

BEGIN

proc\_invalid\_item('-1');

END;

BEGIN

proc\_invalid\_item('i100');

END;

-----------------------------------------------------------------------------------------------------------------------------------------

**REQUIREMENT 7**

SELECT item.itemdesc,item.qty\_hand

FROM TBLOrderDtl od,TBLItem item,TBLOrderMaster om

WHERE od.OrderID = item.ItemID

AND om.orderID = od.OrderID

AND TO\_CHAR(om.odate,"yyyy") = '2014'

AND TO\_CHAR(om.odate,"mon") IN ('sep','oct','nov','dec');

**REQUIREMENT 8**

CREATE OR REPLACE PROCEDURE Proc\_ven\_history (

p\_venid IN VARCHAR2,

p\_vename IN VARCHAR2,

p\_item\_desc IN VARCHAR2

)

AS

BEGIN

INSERT INTO TBLvendor\_history (vencode, venname,Item\_Desc)

VALUES (p\_venid, p\_vename, p\_item\_desc);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLCODE||' '||SQLERRM);

END;

CREATE OR REPLACE PROCEDURE proc\_item(p\_item IN TBLitem.itemdesc%TYPE)

AS

CURSOR c\_ven IS

SELECT TBLV.venid, TBLV.venname, TBLI.itemdesc

FROM TBLvendor TBLV , TBLitem TBLI

WHERE TBLV.venid = TBLI.venid

AND TBLI.itemdesc LIKE p\_item||'%';

rec\_ven c\_ven%ROWTYPE;

plsql\_block VARCHAR2(500);

v\_venid VARCHAR2(10);

v\_vename VARCHAR2(10);

v\_item\_desc VARCHAR2(10);

BEGIN

-- Dynamic PL/SQL block invokes subprogram:

FOR v\_ven\_rec IN c\_ven LOOP

DBMS\_OUTPUT.PUT\_LINE('AAA');

plsql\_block := 'BEGIN Proc\_ven\_history(:venid,:venname,:itemdesc); END;';

EXECUTE IMMEDIATE plsql\_block USING IN v\_ven\_rec.venid, v\_ven\_rec.venname, v\_ven\_rec.itemdesc;

END LOOP;

END;

**TO EXECUTE**

BEGIN

proc\_item('Elect');

END;

**REQUIREMENT 9**

CREATE OR REPLACE PROCEDURE proc\_duplicate\_vendor

(

p\_venid IN TBLVendor.venid%TYPE,

p\_venname IN TBLVendor.venname%TYPE,

p\_venadd1 IN TBLVendor.venadd1%TYPE,

p\_venadd2 IN TBLVendor.venadd2%TYPE,

p\_venadd3 IN TBLVendor.venadd3%TYPE,

p\_number IN TBLVendor.contact\_number%TYPE

)

IS

duplicate\_vendor EXCEPTION;

PRAGMA EXCEPTION\_INIT(duplicate\_vendor,-1);

BEGIN

INSERT INTO TBLVendor VALUES(p\_venid,p\_venname,p\_venadd1,p\_venadd2,p\_venadd3,p\_number);

EXCEPTION

WHEN duplicate\_vendor THEN

DBMS\_OUTPUT.PUT\_LINE('Duplicate Vendor ID'||' '||p\_venid||' '||'should not added in the record');

END;

**TO EXECUTE**

BEGIN

proc\_duplicate\_vendor('v001','James',12,'Nehru Street','Chennai',564789);

END;

**REQUIREMENT 10**

SELECT od.ItemID item\_id, sum(od.Qty\_Ordered) Qty

FROM TBLOrderDtl od, TBLOrderMaster om

WHERE od.orderid = om.orderid

AND TO\_CHAR(om.odate,"yyyy") = '2014'

AND TO\_CHAR(om.odate,"mon") IN ('sep','oct','nov','dec')

GROUP BY od.ItemID