CREATE OR REPLACE PACKAGE BODY MCB.bcm\_main

AS

FUNCTION get\_inv\_ref (p\_ord\_no VARCHAR2)

RETURN VARCHAR2

IS

l\_invoice\_ref VARCHAR2 (100) := ' ';

CURSOR c1 IS

SELECT DISTINCT v.inv\_ref

FROM po\_invoices v

WHERE SUBSTR (v.pol\_ord\_ref, 1, 5) = p\_ord\_no;

BEGIN

FOR i IN c1

LOOP

l\_invoice\_ref := l\_invoice\_ref || i.inv\_ref || ', ';

END LOOP;

RETURN RTRIM (LTRIM (l\_invoice\_ref), ', ');

END;

FUNCTION get\_supp\_first\_contact (p\_supp\_id NUMBER)

RETURN VARCHAR2

IS

CURSOR c1 IS

SELECT REPLACE (REPLACE (s.supp\_contact\_no, ' '), '.') contact\_num

FROM po\_suppliers s

WHERE s.supp\_id = p\_supp\_id;

l\_position NUMBER;

l\_first\_contact\_num VARCHAR2 (10);

BEGIN

FOR i IN c1

LOOP

l\_position := INSTR (i.contact\_num, ',');

IF l\_position = 0

THEN

l\_first\_contact\_num := i.contact\_num;

ELSIF l\_position <> 0

THEN

l\_first\_contact\_num :=

SUBSTR (i.contact\_num, 1, INSTR (i.contact\_num, ',') - 1);

END IF;

IF LENGTH (l\_first\_contact\_num) = 7

THEN -- landline

-- l\_first\_contact\_num :=

--TO\_CHAR (l\_first\_contact\_num, '999-9999');

l\_first\_contact\_num :=

REGEXP\_REPLACE (l\_first\_contact\_num,

'([0-9]{3})([0-9]{4})',

'\1-\2');

ELSIF LENGTH (l\_first\_contact\_num) = 8

THEN -- mobile

--l\_first\_contact\_num :=

--TO\_CHAR (l\_first\_contact\_num, '5999-9999');

l\_first\_contact\_num :=

REGEXP\_REPLACE (l\_first\_contact\_num,

'([0-9]{4})([0-9]{4})',

'\1-\2');

END IF;

END LOOP;

RETURN l\_first\_contact\_num;

END;

FUNCTION get\_supp\_second\_contact (p\_supp\_id NUMBER)

RETURN VARCHAR2

IS

CURSOR c1 IS

SELECT REPLACE (REPLACE (s.supp\_contact\_no, ' '), '.') contact\_num

FROM po\_suppliers s

WHERE s.supp\_id = p\_supp\_id;

l\_position NUMBER;

l\_second\_contact\_num VARCHAR2 (10);

BEGIN

FOR i IN c1

LOOP

l\_position := INSTR (i.contact\_num, ',');

IF l\_position = 0

THEN

l\_second\_contact\_num := NULL;

ELSIF l\_position <> 0

THEN

l\_second\_contact\_num :=

SUBSTR (i.contact\_num,

INSTR (i.contact\_num, ',') + 1,

LENGTH (i.contact\_num));

END IF;

IF LENGTH (l\_second\_contact\_num) = 7

THEN -- landline

l\_second\_contact\_num :=

REGEXP\_REPLACE (l\_second\_contact\_num,

'([0-9]{3})([0-9]{4})',

'\1-\2');

ELSIF LENGTH (l\_second\_contact\_num) = 8

THEN -- mobile

l\_second\_contact\_num :=

REGEXP\_REPLACE (l\_second\_contact\_num,

'([0-9]{4})([0-9]{4})',

'\1-\2');

END IF;

END LOOP;

RETURN l\_second\_contact\_num;

END;

FUNCTION get\_count\_orders (p\_ord\_no VARCHAR2)

RETURN NUMBER

IS

CURSOR c1 IS

SELECT NVL (COUNT (l.pol\_ord\_ref), 0) cnt\_orders

FROM po\_lines l

WHERE SUBSTR (l.pol\_ord\_ref, 1, 5) = p\_ord\_no

AND l.pol\_ord\_line\_status <> 'Cancelled';

BEGIN

FOR i IN c1

LOOP

RETURN i.cnt\_orders;

END LOOP;

RETURN -1;

END;

FUNCTION get\_third\_highest (p\_ord\_no VARCHAR2)

RETURN NUMBER

IS

CURSOR c1 IS

SELECT MIN (x.total\_amt) total\_amt

FROM ( SELECT DISTINCT h.poh\_ord\_total\_amt total\_amt

FROM po\_headers h

ORDER BY h.poh\_ord\_total\_amt DESC) x

WHERE ROWNUM < 4;

BEGIN

FOR i IN c1

LOOP

RETURN i.total\_amt;

END LOOP;

RETURN -1;

END;

FUNCTION get\_po\_serial (p\_order\_no VARCHAR2)

RETURN NUMBER

IS

CURSOR c1 IS

SELECT TO\_NUMBER (SUBSTR (SUBSTR (h.poh\_ord\_no, 1, 5), 3, 5)) po\_serial

FROM po\_headers h

WHERE h.poh\_ord\_no = p\_order\_no;

BEGIN

FOR i IN c1

LOOP

RETURN i.po\_serial;

END LOOP;

RETURN -1;

END;

FUNCTION get\_action (p\_ord\_no VARCHAR2, p\_inv\_status VARCHAR2)

RETURN VARCHAR2

IS

CURSOR c1 IS

SELECT DECODE (UPPER (v.inv\_status),

'PAID', 'OK',

'PENDING', 'TO FOLLOW-UP',

' ', 'TO VERIFY') action\_status

FROM po\_invoices v

WHERE v.pol\_ord\_ref = p\_ord\_no;

BEGIN

FOR i IN c1

LOOP

RETURN i.action\_status;

END LOOP;

RETURN ' ';

END;

FUNCTION get\_amt\_replaced (p\_amt VARCHAR2)

RETURN NUMBER

IS

p\_amt\_replaced NUMBER;

p\_result VARCHAR2 (2);

p\_symbol VARCHAR2 (2);

BEGIN

p\_amt\_replaced :=

TO\_NUMBER (

REPLACE (

REPLACE (REPLACE (REPLACE (p\_amt, ','), 'I', 1), 'o', 0),

'S',

5));

RETURN p\_amt\_replaced;

END;

FUNCTION get\_supp\_id (p\_supp\_name VARCHAR2)

RETURN NUMBER

IS

CURSOR c1 IS

SELECT s.supp\_id

FROM po\_suppliers s

WHERE s.supp\_name = p\_supp\_name;

BEGIN

FOR i IN c1

LOOP

RETURN i.supp\_id;

END LOOP;

RETURN 0;

END;

FUNCTION get\_supplier\_name (p\_supp\_id NUMBER)

RETURN VARCHAR2

IS

CURSOR c1 IS

SELECT s.supp\_name

FROM po\_suppliers s

WHERE s.supp\_id = p\_supp\_id;

BEGIN

FOR i IN c1

LOOP

RETURN i.supp\_name;

END LOOP;

RETURN ' ';

END;

FUNCTION get\_itm\_id (p\_itm\_desc VARCHAR2)

RETURN NUMBER

IS

CURSOR c1 IS

SELECT i.itm\_id

FROM po\_items i

WHERE i.itm\_desc = UPPER (p\_itm\_desc);

BEGIN

FOR i IN c1

LOOP

RETURN i.itm\_id;

END LOOP;

RETURN 0;

END;

PROCEDURE p\_mig\_orders (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2)

IS

CURSOR c1 IS

SELECT t.order\_ref,

TO\_DATE (t.order\_date, 'DD/MM/YYYY')

ord\_dt,

t.supplier\_name,

mcb.bcm\_main.get\_supp\_id (t.supplier\_name)

supp\_id,

TO\_NUMBER (REPLACE (t.order\_total\_amount, ','))

total\_amount,

t.order\_description,

t.order\_status

FROM mcb.xxbcm\_order\_mgt t

---WHERE t.order\_ref LIKE NVL (p\_ord\_no, '%')

WHERE LENGTH (t.order\_ref) = 5

AND NOT EXISTS

(SELECT 1

FROM mcb.po\_headers h

WHERE h.poh\_ord\_no = t.order\_ref)

ORDER BY t.order\_ref;

CURSOR c2 IS

SELECT t.order\_ref,

t.order\_description,

mcb.bcm\_main.get\_itm\_id (t.order\_description)

itm\_id,

t.order\_line\_amount,

mcb.bcm\_main.get\_amt\_replaced (t.order\_line\_amount)

line\_amt,

t.order\_status

FROM mcb.xxbcm\_order\_mgt t

WHERE LENGTH (t.order\_ref) > 5

AND NOT EXISTS

(SELECT 1

FROM po\_lines h

WHERE h.pol\_ord\_ref = t.order\_ref)

ORDER BY t.order\_ref;

CURSOR c3 IS

SELECT t.order\_ref,

mcb.bcm\_main.get\_itm\_id (t.order\_description)

itm\_id,

t.invoice\_reference

invoice\_ref,

TO\_DATE (t.invoice\_date, 'DD/MM/YYYY')

inv\_dt,

t.invoice\_status,

t.invoice\_hold\_reason,

mcb.bcm\_main.get\_amt\_replaced (t.invoice\_amount)

inv\_amount,

t.invoice\_description

FROM mcb.xxbcm\_order\_mgt t

WHERE t.invoice\_reference IS NOT NULL

AND NOT EXISTS

(SELECT 1

FROM mcb.po\_invoices v

WHERE v.inv\_ref = t.invoice\_reference

AND v.pol\_ord\_ref = t.order\_ref)

ORDER BY t.invoice\_reference;

BEGIN

-- Initialisation

p\_return\_status := 'S';

IF p\_return\_status = 'S'

THEN

-- insert headers

BEGIN

FOR i IN c1

LOOP

mcb.bcm\_main.insert\_po\_headers (

p\_return\_status => p\_return\_status,

p\_msg\_data => p\_msg\_data,

p\_type => 'I',

p\_ord\_no => i.order\_ref,

p\_ord\_dt => i.ord\_dt,

p\_supp\_id => i.supp\_id,

p\_ord\_total\_amt => i.total\_amount,

p\_ord\_desc => i.order\_description,

p\_ord\_status => i.order\_status);

END LOOP;

-- insert lines

FOR j IN c2

LOOP

mcb.bcm\_main.insert\_po\_lines (

p\_return\_status => p\_return\_status,

p\_msg\_data => p\_msg\_data,

p\_type => 'I',

p\_ord\_no => SUBSTR (j.order\_ref, 1, 5),

p\_ord\_ref => j.order\_ref,

p\_itm\_id => j.itm\_id,

p\_ord\_line\_status => j.order\_status,

p\_ord\_line\_amt => j.line\_amt);

END LOOP;

-- insert invoices

FOR k IN c3

LOOP

mcb.bcm\_main.insert\_po\_invoices (

p\_return\_status => p\_return\_status,

p\_msg\_data => p\_msg\_data,

p\_type => 'I',

p\_inv\_ref => k.invoice\_ref,

p\_inv\_dt => k.inv\_dt,

p\_ord\_ref => k.order\_ref,

p\_itm\_id => k.itm\_id,

p\_inv\_status => k.invoice\_status,

p\_inv\_hold\_reason => k.invoice\_hold\_reason,

p\_inv\_amt => k.inv\_amount,

p\_inv\_desc => k.invoice\_description);

END LOOP;

COMMIT;

p\_msg\_data := 'Processing completed';

DBMS\_OUTPUT.put\_line (p\_msg\_data);

END;

END IF;

EXCEPTION

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error @ ' || SQLERRM;

END;

PROCEDURE p\_mig\_supplier (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2)

IS

CURSOR c1 IS

SELECT DISTINCT t.supplier\_name,

t.supp\_contact\_name,

t.supp\_address,

t.supp\_contact\_number,

t.supp\_email

FROM mcb.xxbcm\_order\_mgt t

ORDER BY 1;

BEGIN

-- Initialisation

p\_return\_status := 'S';

IF p\_return\_status = 'S'

THEN

BEGIN

FOR i IN c1

LOOP

mcb.bcm\_main.insert\_po\_suppliers (

p\_return\_status => p\_return\_status,

p\_msg\_data => p\_msg\_data,

p\_type => 'I',

p\_supp\_name => i.supplier\_name,

p\_supp\_contact\_name => i.supp\_contact\_name,

p\_supp\_address => i.supp\_address,

p\_supp\_contact\_no => i.supp\_contact\_number,

p\_supp\_mobile\_no => NULL,

p\_supp\_landline\_no => NULL,

p\_supp\_email => i.supp\_email);

END LOOP;

COMMIT;

p\_msg\_data := 'Processing completed';

DBMS\_OUTPUT.put\_line (p\_msg\_data);

END;

END IF;

EXCEPTION

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error @ ' || SQLERRM;

END;

PROCEDURE p\_mig\_items (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2)

IS

CURSOR c1 IS

SELECT DISTINCT (t.order\_description) item\_desc

FROM mcb.xxbcm\_order\_mgt t

WHERE t.order\_line\_amount IS NOT NULL

--WHERE LENGTH (t.order\_ref) > 5

ORDER BY t.order\_description;

BEGIN

-- Initialisation

p\_return\_status := 'S';

IF p\_return\_status = 'S'

THEN

BEGIN

FOR i IN c1

LOOP

mcb.bcm\_main.insert\_po\_items (

p\_return\_status => p\_return\_status,

p\_msg\_data => p\_msg\_data,

p\_type => 'I',

p\_itm\_desc => UPPER (i.item\_desc));

END LOOP;

COMMIT;

p\_msg\_data := 'Processing completed';

DBMS\_OUTPUT.put\_line (p\_msg\_data);

END;

END IF;

EXCEPTION

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error @ ' || SQLERRM;

END;

PROCEDURE insert\_po\_headers (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2,

p\_type VARCHAR2,

p\_ord\_no VARCHAR2,

p\_ord\_dt DATE,

p\_supp\_id NUMBER,

p\_ord\_total\_amt NUMBER,

p\_ord\_desc VARCHAR2,

p\_ord\_status VARCHAR2)

IS

BEGIN

p\_return\_status := 'S';

BEGIN

IF p\_return\_status = 'S' AND p\_type = 'I'

THEN

INSERT INTO mcb.po\_headers (poh\_ord\_no,

poh\_ord\_dt,

supp\_id,

poh\_ord\_total\_amt,

poh\_ord\_desc,

poh\_ord\_status)

VALUES (p\_ord\_no,

p\_ord\_dt,

p\_supp\_id,

p\_ord\_total\_amt,

p\_ord\_desc,

p\_ord\_status);

END IF;

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX

THEN

--p\_return\_status := 'F';

INSERT INTO tmp\_ins\_msg (msg, entity)

VALUES (

'Duplicate value for Order found -'

|| p\_ord\_no,

'BCM\_ORD');

p\_msg\_data := 'Duplicate value for Order found -' || p\_ord\_no;

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error occured @' || SQLERRM;

END;

END;

PROCEDURE insert\_po\_lines (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2,

p\_type VARCHAR2,

p\_ord\_no VARCHAR2,

p\_ord\_ref VARCHAR2,

p\_itm\_id NUMBER,

p\_ord\_line\_status VARCHAR2,

p\_ord\_line\_amt NUMBER)

IS

l\_line\_no NUMBER;

BEGIN

p\_return\_status := 'S';

BEGIN

IF p\_return\_status = 'S' AND p\_type = 'I'

THEN

SELECT NVL (MAX (l.pol\_ord\_line\_seq), 0) + 10

INTO l\_line\_no

FROM mcb.po\_lines l

WHERE l.pol\_ord\_ref = p\_ord\_ref;

INSERT INTO mcb.po\_lines (poh\_ord\_no,

pol\_ord\_ref,

pol\_ord\_line\_seq,

itm\_id,

pol\_ord\_line\_status,

pol\_ord\_line\_amt)

VALUES (p\_ord\_no,

p\_ord\_ref,

l\_line\_no,

p\_itm\_id,

p\_ord\_line\_status,

p\_ord\_line\_amt);

END IF;

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX

THEN

--p\_return\_status := 'F';

INSERT INTO tmp\_ins\_msg (msg, entity)

VALUES (

'Duplicate value for Order Line found -'

|| p\_ord\_ref,

'BCM\_ORD\_LINE');

p\_msg\_data :=

'Duplicate value for Order Line found -' || p\_ord\_ref;

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error occured @' || SQLERRM;

END;

END;

PROCEDURE insert\_po\_invoices (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2,

p\_type VARCHAR2,

p\_inv\_ref VARCHAR2,

p\_inv\_dt DATE,

p\_ord\_ref VARCHAR2,

p\_itm\_id NUMBER,

p\_inv\_status VARCHAR2,

p\_inv\_hold\_reason VARCHAR2,

p\_inv\_amt NUMBER,

p\_inv\_desc VARCHAR2)

IS

BEGIN

p\_return\_status := 'S';

BEGIN

IF p\_return\_status = 'S' AND p\_type = 'I'

THEN

INSERT INTO mcb.po\_invoices (inv\_ref,

inv\_dt,

pol\_ord\_ref,

itm\_id,

inv\_status,

inv\_hold\_reason,

inv\_amt,

inv\_desc)

VALUES (p\_inv\_ref,

p\_inv\_dt,

p\_ord\_ref,

p\_itm\_id,

p\_inv\_status,

p\_inv\_hold\_reason,

p\_inv\_amt,

p\_inv\_desc);

END IF;

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX

THEN

--p\_return\_status := 'F';

INSERT INTO tmp\_ins\_msg (msg, entity)

VALUES (

'Duplicate value for invoice found -'

|| p\_inv\_ref,

'BCM\_INVOICES');

p\_msg\_data :=

'Duplicate value for invoice found -' || p\_inv\_ref;

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error occured @' || SQLERRM;

END;

END;

PROCEDURE insert\_po\_suppliers (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2,

p\_type VARCHAR2,

p\_supp\_name VARCHAR2,

p\_supp\_contact\_name VARCHAR2,

p\_supp\_address VARCHAR2,

p\_supp\_contact\_no VARCHAR2,

p\_supp\_mobile\_no NUMBER,

p\_supp\_landline\_no NUMBER,

p\_supp\_email VARCHAR2)

IS

l\_seq\_no NUMBER;

BEGIN

p\_return\_status := 'S';

BEGIN

IF p\_return\_status = 'S' AND p\_type = 'I'

THEN

SELECT NVL (MAX (s.supp\_id), 100) + 1

INTO l\_seq\_no

FROM mcb.po\_suppliers s;

INSERT INTO mcb.po\_suppliers (supp\_id,

supp\_name,

supp\_contact\_name,

supp\_address,

supp\_contact\_no,

supp\_mobile\_no,

supp\_landline\_no,

supp\_email)

VALUES (l\_seq\_no,

p\_supp\_name,

p\_supp\_contact\_name,

p\_supp\_address,

p\_supp\_contact\_no,

p\_supp\_mobile\_no,

p\_supp\_landline\_no,

p\_supp\_email);

END IF;

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX

THEN

--p\_return\_status := 'F';

INSERT INTO tmp\_ins\_msg (msg, entity)

VALUES (

'Duplicate value for supplier found -'

|| p\_supp\_name,

'BCM\_SUPP');

p\_msg\_data :=

'Duplicate value for supplier found -' || p\_supp\_name;

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error occured @' || SQLERRM;

END;

END;

PROCEDURE insert\_po\_items (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2,

p\_type VARCHAR2,

p\_itm\_desc VARCHAR2)

IS

l\_seq\_no NUMBER;

BEGIN

p\_return\_status := 'S';

BEGIN

IF p\_return\_status = 'S' AND p\_type = 'I'

THEN

SELECT NVL (MAX (i.itm\_id), 1000) + 1

INTO l\_seq\_no

FROM mcb.po\_items i;

INSERT INTO mcb.po\_items (itm\_id, itm\_desc)

VALUES (l\_seq\_no, p\_itm\_desc);

END IF;

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX

THEN

--p\_return\_status := 'F';

INSERT INTO tmp\_ins\_msg (msg, entity)

VALUES (

'Duplicate value for item found -'

|| p\_itm\_desc,

'BCM\_ITM');

p\_msg\_data :=

'Duplicate value for item found -' || p\_itm\_desc;

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error occured @' || SQLERRM;

END;

END;

PROCEDURE p\_mig\_upd\_supplier (p\_return\_status OUT VARCHAR2,

p\_msg\_data OUT VARCHAR2)

IS

CURSOR c1 IS

SELECT s.supp\_id,

REPLACE (REPLACE (s.supp\_contact\_no, ' '), '.') contact\_num

FROM po\_suppliers s

ORDER BY s.supp\_id;

CURSOR c2 IS

SELECT s.supp\_id,

REPLACE (REPLACE (s.supp\_contact\_no, ' '), '.') contact\_num

FROM po\_suppliers s

ORDER BY s.supp\_id;

l\_position NUMBER;

l\_first\_contact\_num VARCHAR2 (10);

l\_second\_contact\_num VARCHAR2 (10);

BEGIN

-- Initialisation

p\_return\_status := 'S';

IF p\_return\_status = 'S'

THEN

BEGIN

FOR i IN c1

LOOP

l\_position := INSTR (i.contact\_num, ',');

IF l\_position = 0

THEN

l\_first\_contact\_num := i.contact\_num;

ELSIF l\_position <> 0

THEN

l\_first\_contact\_num :=

SUBSTR (i.contact\_num,

1,

INSTR (i.contact\_num, ',') - 1);

END IF;

IF LENGTH (l\_first\_contact\_num) = 7

THEN -- landline

UPDATE po\_suppliers s

SET s.supp\_landline\_no =

TO\_NUMBER (l\_first\_contact\_num)

WHERE supp\_id = i.supp\_id;

ELSIF LENGTH (l\_first\_contact\_num) = 8

THEN -- mobile

UPDATE po\_suppliers s

SET s.supp\_mobile\_no =

TO\_NUMBER (l\_first\_contact\_num)

WHERE supp\_id = i.supp\_id;

END IF;

END LOOP;

FOR j IN c2

LOOP

l\_position := INSTR (j.contact\_num, ',');

IF l\_position = 0

THEN

l\_second\_contact\_num := NULL;

ELSIF l\_position <> 0

THEN

l\_second\_contact\_num :=

SUBSTR (j.contact\_num,

INSTR (j.contact\_num, ',') + 1,

LENGTH (j.contact\_num));

END IF;

IF LENGTH (l\_second\_contact\_num) = 7

THEN -- landline

UPDATE po\_suppliers s

SET s.supp\_landline\_no =

TO\_NUMBER (l\_second\_contact\_num)

WHERE supp\_id = j.supp\_id;

ELSIF LENGTH (l\_second\_contact\_num) = 8

THEN -- mobile

UPDATE po\_suppliers s

SET s.supp\_mobile\_no =

TO\_NUMBER (l\_second\_contact\_num)

WHERE supp\_id = j.supp\_id;

END IF;

END LOOP;

COMMIT;

p\_msg\_data := 'Processing completed';

DBMS\_OUTPUT.put\_line (p\_msg\_data);

END;

END IF;

EXCEPTION

WHEN OTHERS

THEN

p\_return\_status := 'F';

p\_msg\_data := 'Error @ ' || SQLERRM;

END;

END;

/