# Step 1: Create tables

CREATE TABLE SUPPLIERS (

SUPPLIER\_ID NUMBER(3) NOT NULL,

SUPPLIER\_NAME VARCHAR2(50) NOT NULL,

CONTACT\_NAME VARCHAR2(50) NOT NULL,

ADDRESS VARCHAR2(200) NOT NULL,

CONTACT\_NUMBER VARCHAR2(50) NOT NULL,

EMAIL VARCHAR2(50),

CREATED\_DATE DATE,

CREATED\_BY VARCHAR2(10),

CONSTRAINT SUPP\_PK PRIMARY KEY ( SUPPLIER\_ID )

);

CREATE TABLE ORDERS (

ORDER\_REF VARCHAR2(5) NOT NULL,

SUPPLIER\_ID NUMBER(3) NOT NULL,

ORDER\_DATE DATE,

TOTAL\_AMOUNT NUMBER(10),

ORD\_DESCRIPTION VARCHAR2(200),

ORD\_STATUS VARCHAR2(10),

CREATED\_DATE DATE,

CREATED\_BY VARCHAR2(10),

CONSTRAINT ORDERS\_PK PRIMARY KEY ( ORDER\_REF )

);

CREATE TABLE ORDER\_LINES (

ORD\_LINE\_SEQ NUMBER(5) NOT NULL,

ORDER\_REF VARCHAR2(5) NOT NULL,

ORDER\_REF\_LINE\_ID NUMBER(2),

LINE\_DESCRIPTION VARCHAR2(200),

LINE\_STATUS VARCHAR2(10),

LINE\_AMOUNT NUMBER(10),

INV\_SEQ NUMBER(5),

INV\_REFERENCE VARCHAR2(9),

CREATED\_DATE DATE,

CREATED\_BY VARCHAR2(10),

CONSTRAINT ORDER\_LINES\_PK PRIMARY KEY ( ORD\_LINE\_SEQ,

ORDER\_REF,

ORDER\_REF\_LINE\_ID )

);

CREATE TABLE INVOICES (

INV\_SEQ NUMBER(5) NOT NULL,

INV\_REFERENCE VARCHAR2(9) NOT NULL,

INV\_REF\_PAY NUMBER(3),

INV\_DATE DATE,

INV\_STATUS VARCHAR2(10),

INV\_HOLD\_REASON VARCHAR2(100),

INV\_AMOUNT NUMBER(10),

INV\_DESCRIPTION VARCHAR2(200),

INV\_CREATED\_DATE DATE,

INV\_CREATED\_BY VARCHAR2(10),

CONSTRAINT INVOICES\_PK PRIMARY KEY ( INV\_SEQ,

INV\_REFERENCE )

);

ALTER TABLE ORDERS

ADD CONSTRAINT SUPP\_ID\_FK\_ORDERS

FOREIGN KEY (SUPPLIER\_ID)

REFERENCES SUPPLIERS (SUPPLIER\_ID);

ALTER TABLE ORDER\_LINES

ADD CONSTRAINT ORDER\_REF\_FK\_ORDER\_LINES

FOREIGN KEY (ORDER\_REF)

REFERENCES ORDERS (ORDER\_REF);

ALTER TABLE ORDER\_LINES

ADD CONSTRAINT INV\_SEQ\_FK\_ORDER\_LINES

FOREIGN KEY (INV\_SEQ,INV\_REFERENCE)

REFERENCES INVOICES (INV\_SEQ,INV\_REFERENCE);

# Step 2: Create package: ord\_pck and compile

Header:

create or replace package ord\_pck as

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure clear\_all\_tables;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_amount\_number(p\_amount in varchar2) return number;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_date\_from\_string(p\_date in varchar2) return date ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_payment\_from\_inv(p\_invoice\_reference in varchar2) return number ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure create\_order\_lines(p\_order\_ref in varchar2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure create\_invoices(p\_invoice\_reference in varchar2,p\_invoice\_date in date,p\_invoice\_status in varchar2,

p\_invoice\_hold\_reason in varchar2,p\_invoice\_amount in number,p\_invoice\_description in varchar2,p\_inv\_seq out number,

p\_inv\_reference out varchar2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure create\_suppliers ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure migration\_process;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure distinct\_invoices\_with\_total ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure get\_nth\_highest\_order\_amount(p\_number in number);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_invoices(p\_order\_ref in varchar2) return varchar2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure list\_suppliers\_with\_orders(p\_date\_from in date,p\_date\_to in date) ;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure get\_detailed\_contact(p\_contact\_number in varchar2,p\_out\_contact1 out varchar2,p\_out\_contact2 out varchar2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_action\_status (p\_order\_ref in varchar2) return varchar2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

end ord\_pck;

Package body:

create or replace PACKAGE BODY ord\_pck AS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ASEESURUN Package to import from XXBCM\_ORDER\_MGT raw format data into their respective tables

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE clear\_all\_tables AS

--This procedure is used to clear all the tables before the migration occurs.

BEGIN

DELETE FROM order\_lines;

DELETE FROM orders;

DELETE FROM invoices;

DELETE FROM suppliers;

COMMIT;

END clear\_all\_tables;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FUNCTION get\_amount\_number (p\_amount IN VARCHAR2) RETURN NUMBER AS

--This function will take the varchar2 and convert into number

--Replace , by '' and 'S' by 5 and O by 0 and I by 1 for typo errors

m\_amount NUMBER ;

BEGIN

if p\_amount is not null then

SELECT

to\_number(replace(replace(replace(replace(upper(p\_amount), ',', ''), 'S', 5), 'O', 0), 'I', 1))

INTO m\_amount

FROM

dual;

end if;

return(m\_amount);

END get\_amount\_number;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FUNCTION get\_date\_from\_string (p\_date IN VARCHAR2) RETURN DATE AS

--This function will reformat the date accordingly and return it into a date format

--The file contains only two formats, DD-MON-YYYY and DD-MM-YYYY

--More formats can be added

BEGIN

IF p\_date IS NOT NULL THEN

BEGIN

RETURN to\_date(p\_date, 'DD-MON-YYYY');

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

BEGIN

RETURN to\_date(p\_date, 'DD-MM-YYYY');

EXCEPTION

WHEN OTHERS THEN

NULL;

END;

END IF;

RETURN NULL;

END get\_date\_from\_string;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_payment\_from\_inv(p\_invoice\_reference in varchar2) return number as

m\_inv\_ref\_pay number:=0;

begin

if length(p\_invoice\_reference)> 9 then

--therefore it contains a payment number

select to\_number(replace(p\_invoice\_reference,substr(p\_invoice\_reference, 1, 9)||'.',''))

into m\_inv\_ref\_pay

from dual;

return(m\_inv\_ref\_pay);

else --it does not have any payment number return null

return(null);

end if;

end get\_payment\_from\_inv;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE create\_suppliers AS

CURSOR c\_suppliers IS

SELECT DISTINCT

supplier\_name,

supp\_contact\_name,

supp\_address,

replace(replace(replace(replace(replace(TRIM(upper(supp\_contact\_number)), 'O', 0), 'S', 5), ' ', ''), '.', ''), 'I', 1

) contact\_number,

supp\_email

FROM

xxbcm\_order\_mgt

ORDER BY

supplier\_name;

m\_cnt NUMBER := 0;

BEGIN

FOR r1 IN c\_suppliers LOOP

m\_cnt := m\_cnt + 1;

INSERT INTO suppliers VALUES (

m\_cnt,

r1.supplier\_name,

r1.supp\_contact\_name,

r1.supp\_address,

r1.contact\_number,

r1.supp\_email,

sysdate,

user

);

END LOOP;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

raise\_application\_error(-20201, sqlerrm(sqlcode));

END create\_suppliers;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE create\_orders AS

CURSOR c\_ord IS

SELECT

order\_ref,

b.supplier\_id,

ord\_pck.get\_date\_from\_string(a.order\_date) order\_date,

a.order\_total\_amount,

a.order\_description,

a.order\_status

FROM

xxbcm\_order\_mgt a,

suppliers b

WHERE

a.order\_line\_amount IS NULL

AND a.supplier\_name = b.supplier\_name

ORDER BY

a.order\_ref;

BEGIN

FOR r1 IN c\_ord LOOP

INSERT INTO orders (

order\_ref,

supplier\_id,

order\_date,

total\_amount,

ord\_description,

ord\_status,

created\_date,

created\_by

) VALUES (

r1.order\_ref,

r1.supplier\_id,

r1.order\_date,

get\_amount\_number(r1.order\_total\_amount),

r1.order\_description,

r1.order\_status,

sysdate,

user

);

create\_order\_lines(r1.order\_ref);

END LOOP;

COMMIT;

END;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE create\_order\_lines (

p\_order\_ref IN VARCHAR2

) AS

CURSOR c\_ord\_lines IS

SELECT

a.order\_ref order\_ref\_line\_id,

a.order\_description line\_description,

order\_status line\_status,

ord\_pck.get\_amount\_number(order\_line\_amount) line\_amount,

invoice\_reference,

invoice\_date,

invoice\_status,

invoice\_hold\_reason,

invoice\_amount,

invoice\_description

FROM

xxbcm\_order\_mgt a

WHERE

order\_line\_amount IS NOT NULL

AND substr(a.order\_ref, 1, 5) = p\_order\_ref

ORDER BY

order\_ref;

m\_cnt\_seq NUMBER := 1;

m\_inv\_seq invoices.inv\_seq%type;

m\_inv\_reference invoices.inv\_reference%type;

BEGIN

FOR r1 IN c\_ord\_lines LOOP

INSERT INTO order\_lines (

ord\_line\_seq,

order\_ref,

order\_ref\_line\_id,

line\_description,

line\_status,

line\_amount,

created\_date,

created\_by

) VALUES (

m\_cnt\_seq,

p\_order\_ref,

to\_number(replace(r1.order\_ref\_line\_id, p\_order\_ref || '-', '')),

r1.line\_description,

r1.line\_status,

r1.line\_amount,

sysdate,

user

);

--create the invoice

if r1.invoice\_reference is not null then

create\_invoices (r1.invoice\_reference,get\_date\_from\_string(r1.invoice\_date),r1.invoice\_status,r1.invoice\_hold\_reason,

ord\_pck.get\_amount\_number(r1.invoice\_amount),r1.invoice\_description,m\_inv\_seq,m\_inv\_reference);

update order\_lines

set inv\_seq=m\_inv\_seq,inv\_reference=m\_inv\_reference

where ord\_line\_seq=m\_cnt\_seq

and order\_ref=p\_order\_ref;

end if;

m\_cnt\_seq := m\_cnt\_seq + 1;

END LOOP;

END create\_order\_lines;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE create\_invoices (

p\_invoice\_reference IN VARCHAR2,

p\_invoice\_date IN DATE,

p\_invoice\_status IN VARCHAR2,

p\_invoice\_hold\_reason IN VARCHAR2,

p\_invoice\_amount IN NUMBER,

p\_invoice\_description IN VARCHAR2,

p\_inv\_seq OUT NUMBER,

p\_inv\_reference OUT VARCHAR2

) AS

m\_inv\_seq NUMBER := 0;

BEGIN

--check if invoice\_exists in invoice table

SELECT

nvl(MAX(inv\_seq),0)

INTO m\_inv\_seq

FROM

invoices

WHERE

inv\_reference = substr(p\_invoice\_reference, 1, 9);

--add 1 before insert

m\_inv\_seq := m\_inv\_seq + 1;

INSERT INTO invoices (

inv\_seq,

inv\_reference,

inv\_ref\_pay,

inv\_date,

inv\_status,

inv\_hold\_reason,

inv\_amount,

inv\_description,

inv\_created\_date,

inv\_created\_by

) VALUES (

m\_inv\_seq,

substr(p\_invoice\_reference, 1, 9),

get\_payment\_from\_inv(p\_invoice\_reference),

p\_invoice\_date,

p\_invoice\_status,

p\_invoice\_hold\_reason,

p\_invoice\_amount,

p\_invoice\_description,

sysdate,

user

);

p\_inv\_seq:=m\_inv\_seq;

p\_inv\_reference:=substr(p\_invoice\_reference, 1, 9);

END create\_invoices;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PROCEDURE migration\_process AS

BEGIN

dbms\_output.put\_line('Start migration process');

clear\_all\_tables;

dbms\_output.put\_line('All tables cleared');

create\_suppliers;

dbms\_output.put\_line('All suppliers created');

create\_orders;

dbms\_output.put\_line('Orders successfully created');

dbms\_output.put\_line('Migration over');

exception when others then

raise\_application\_error(-20201,sqlerrm(sqlcode));

END migration\_process;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure distinct\_invoices\_with\_total as

--summary of Orders with their corresponding list of distinct invoices and their total amount

cursor c\_results is

SELECT

a.order\_ref,

to\_number(substr(a.order\_ref, - 3, 5)) ord\_ref,

to\_char(a.order\_date, 'MON-YY') order\_period,

initcap(b.supplier\_name) supp\_name,

to\_char(a.total\_amount, '99,999,990.00') tot\_amt,

ord\_status ord\_status,

c.inv\_reference inv\_ref,

to\_char(c.invoice\_amount, '99,999,990.00') inv\_amt

FROM

orders a,

suppliers b,

(

SELECT DISTINCT

( inv\_reference ),

SUM(inv\_amount) invoice\_amount

FROM

invoices

GROUP BY

inv\_reference

) c

WHERE

a.supplier\_id = b.supplier\_id

AND substr(c.inv\_reference, 5, 5) = a.order\_ref

ORDER BY

order\_date DESC;

m\_action varchar2(25);

begin

dbms\_output.put\_line('-----------------------------------------------LIST OF DISTINCT INVOICES AND THEIR TOTAL AMOUNT------------------------------------------------------------------------------');

dbms\_output.put\_line('Order Reference Order Period Supplier Name Order Total Amount Order Status Invoice Reference Invoice Total Amount Action');

dbms\_output.put\_line('------------------------------------------------------------------------------------------------------------------------------------------------------------------------------');

for r1 in c\_results loop

m\_action:=get\_action\_status(r1.order\_ref);

dbms\_output.put\_line(RPAD(r1.ord\_ref,18)||RPAD(r1.order\_period,13)||RPAD(r1.supp\_name,30)||RPAD(r1.tot\_amt,22)||RPAD(r1.ord\_status,17)||rpad(r1.inv\_ref,20)||rpad(r1.inv\_amt,22)||m\_action);

dbms\_output.put\_line('------------------------------------------------------------------------------------------------------------------------------------------------------------------------------');

end loop;

end distinct\_invoices\_with\_total;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure get\_nth\_highest\_order\_amount(p\_number in number) as

--Return details for the Nth highest Order Total Amount from the list.

cursor c\_output is

select a.order\_ref,upper(b.supplier\_name) supp\_name,to\_char(a.order\_date,'fmMonth DD,YYYY') ord\_date,a.ord\_status,to\_char(a.total\_amount,'99,999,990.00') amt

from orders a,suppliers b

where a.supplier\_id=b.supplier\_id

order by total\_amount desc;

m\_count number:=0;

m\_invref varchar2(500);

begin

for r1 in c\_output loop

m\_count:=m\_count+1;

if m\_count=p\_number then

--get all invoices for that order

m\_invref:=get\_invoices(r1.order\_ref);

dbms\_output.put\_line('-----------------------------------------------'||to\_char(to\_date(p\_number,'DD'),'DDSPTH')||' HIGHEST ORDER TOTAL AMOUNT----------------------------------------------------------');

dbms\_output.put\_line('Order Reference Order Date Supplier Name Order Total Amount Order Status Invoice References');

dbms\_output.put\_line('--------------------------------------------------------------------------------------------------------------------------------------------------');

dbms\_output.put\_line(RPAD(r1.order\_ref,18)||RPAD(r1.ord\_date,21)||RPAD(r1.supp\_name,20)||RPAD(r1.amt,25)||RPAD(r1.ord\_status,15)||m\_invref);

exit;

end if;

end loop;

end get\_nth\_highest\_order\_amount;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_invoices(p\_order\_ref in varchar2) return varchar2 as

--Get list of invoices per order

cursor c1 is

select distinct(decode(a.inv\_ref\_pay,null,a.inv\_reference,a.inv\_reference||'.'||a.inv\_ref\_pay)) inv

from invoices a,order\_lines b

where a.inv\_seq=b.inv\_seq

and a.inv\_reference=b.inv\_reference

and b.order\_ref=p\_order\_ref;

m\_invoicerefs varchar2(500);

begin

for r1 in c1 loop

if m\_invoicerefs is null then

m\_invoicerefs :=r1.inv;

else

m\_invoicerefs :=m\_invoicerefs||','||r1.inv;

end if;

end loop;

return(m\_invoicerefs);

end get\_invoices;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure list\_suppliers\_with\_orders(p\_date\_from in date,p\_date\_to in date) as

--List all suppliers with their respective number of orders and total amount ordered from them between any period

cursor c\_suppliers is

select a.supplier\_name,a.contact\_name,contact\_number,count('\*') total\_orders,to\_char(sum(b.total\_amount),'99,999,990.00') total\_amt

from suppliers a,orders b

where a.supplier\_id=b.supplier\_id

and b.order\_date between p\_date\_from and p\_date\_to

group by a.supplier\_name,a.contact\_name,contact\_number;

m\_contact1 varchar2(15);

m\_contact2 varchar2(15);

begin

dbms\_output.put\_line('--------------------NUMBER OF ORDERS AND TOTAL AMOUNT ORDERED FROM PERIOD '||p\_date\_from||' to '||p\_date\_to||'----------------------------------------------------------');

dbms\_output.put\_line('--------------------------------------------------------------------------------------------------------------------------------------------------');

dbms\_output.put\_line(' Supplier Name Supplier Contact Name Supplier Contact No.1 Supplier Contact No.2 Total Orders Order Total Amount');

for r1 in c\_suppliers loop

get\_detailed\_contact(r1.contact\_number,m\_contact1,m\_contact2);

dbms\_output.put\_line('--------------------------------------------------------------------------------------------------------------------------------------------------');

dbms\_output.put\_line(RPAD(r1.supplier\_name,30)||RPAD(r1.contact\_name,25)||RPAD(m\_contact1,25)||RPAD(m\_contact2,25)||RPAD(r1.total\_orders,10)||r1.total\_amt);

end loop;

end list\_suppliers\_with\_orders;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Procedure get\_detailed\_contact(p\_contact\_number in varchar2,p\_out\_contact1 out varchar2,p\_out\_contact2 out varchar2) as

--This procedure will check if there is a , in the phone number

--It will then extract the phone numbers and format them accordingly.

m\_contact1 varchar2(15):='';

m\_contact2 varchar2(15):='';

begin

if instr(p\_contact\_number,',')=0 then

m\_contact1:=p\_contact\_number;

m\_contact2:=' ';

else

m\_contact1:=replace(substr(p\_contact\_number,1,instr(p\_contact\_number,',')),',','');

m\_contact2:=replace(substr(p\_contact\_number,instr(p\_contact\_number,','),length(p\_contact\_number)),',','');

end if;

if length(m\_contact1) = 7 then

m\_contact1:=to\_char(substr(m\_contact1, 1, 3) || '-' || substr(m\_contact1, 4, 4));

elsif length(m\_contact1)=8 then

m\_contact1:=to\_char(substr(m\_contact1, 1, 4) || '-' || substr(m\_contact1, 4, 5));

else

m\_contact1:=' ';

end if;

if length(m\_contact2) = 7 then

m\_contact2:=to\_char(substr(m\_contact2, 1, 3) || '-' || substr(m\_contact2, 4, 4));

elsif length(m\_contact2)=8 then

m\_contact2:=to\_char(substr(m\_contact2, 1, 4) || '-' || substr(m\_contact2, 4, 5));

else

m\_contact2:=' ';

end if;

p\_out\_contact1:=m\_contact1;

p\_out\_contact2:=m\_contact2;

end get\_detailed\_contact;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Function get\_action\_status (p\_order\_ref in varchar2) return varchar2 as

m\_action varchar2(25);

m\_blank number:=0;

m\_paid number:=0;

m\_pending number:=0;

begin

select count('\*')

into m\_blank

from order\_lines

where order\_ref=p\_order\_ref

and inv\_reference is null; --This means there is no invoice attached to this line, status is blank

if m\_blank >0 then

return('To verify');

else

select sum(decode(upper(inv\_status),'PAID',1,0)) PAID,sum(decode(upper(inv\_status),'PENDING',1,0)) PENDING

into m\_paid ,m\_pending

from invoices

where substr(inv\_reference,5,5) = p\_order\_ref;

end if;

if m\_pending>0 then --if any invoice is pending then

return('To follow up');

else --all must thave been paid

return('OK');

end if;

return(m\_action);

end get\_action\_status;

END ord\_pck;

# Step 3:

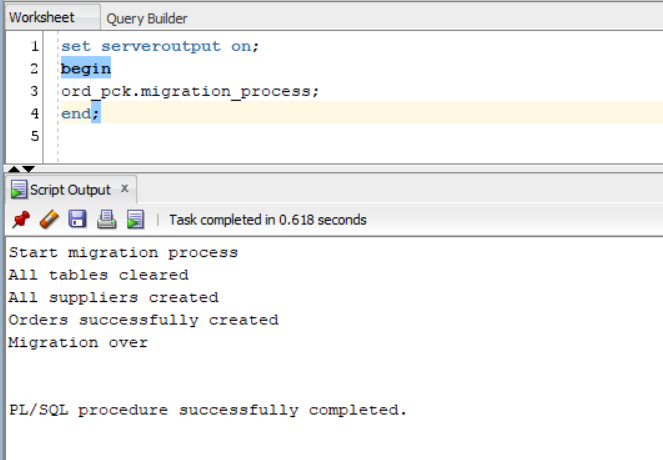
Migration process:

set serveroutput on;

begin

ord\_pck.migration\_process;

end;



4.The owner wishes to have a report displaying a summary of Orders with their corresponding list of ***distinct*** ***invoices and their total amount*** to be able to reconcile his orders and payments. The report shall contain the details as per table below ordered by latest *Order Date* on top. Implement a Stored Procedure or Function to return the required information.

|  |  |
| --- | --- |
| **Column Name** | **Specifications** |
| Order Reference | Exclude prefix PO and return only numeric value, e.g. PO001 shall return value 1. |
| Order Period | Period based on *Order Date*. Example of expected format “JAN-17”. |
| Supplier Name | First character in each word to uppercase and the rest to lowercase. |
| Order Total Amount | Format “99,999,990.00”. |
| Order Status | As per record. |
| Invoice Reference | As per record. |
| Invoice Total Amount | Format “99,999,990.00”. |
| Action | Return the following value based on the invoice statuses.  If all invoice statuses are Paid, display “OK”.  If any of the invoice statuses is Pending, display “To follow up”.  If any of the invoice Statuses is Blank, display “To verify”. |

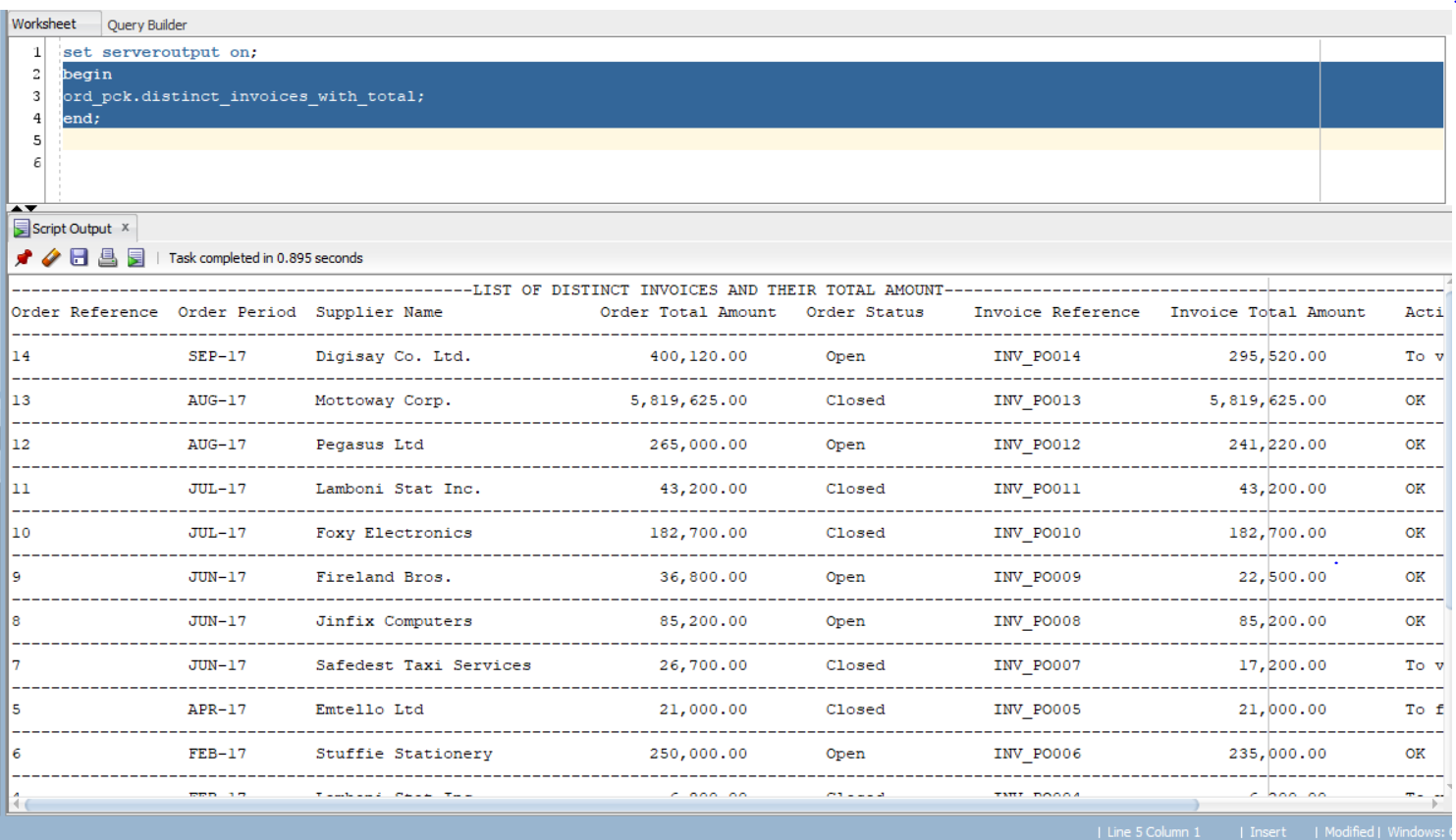
Answer:

set serveroutput on;

begin

ord\_pck.distinct\_invoices\_with\_total;

end;



5.Return details for the ***THIRD (3rd) highest Order Total Amount*** from the list. Only one record is expected with the following information. Implement a Stored Procedure or Function to return the required information.

|  |  |
| --- | --- |
| **Column Name** | **Specifications** |
| Order Reference | Exclude prefix PO and return only numeric value, e.g. PO001 shall return value 1. |
| Order Date | Example of expected format “January 01, 2017”. |
| Supplier Name | In upper case. |
| Order Total Amount | Format “99,999,990.00”. |
| Order Status | As per record. |
| Invoice References | For that specific Order, list all invoice references *delimited by comma*. For example., “INV\_PO999.1, INV\_PO999.2, INV\_PO999.3” |

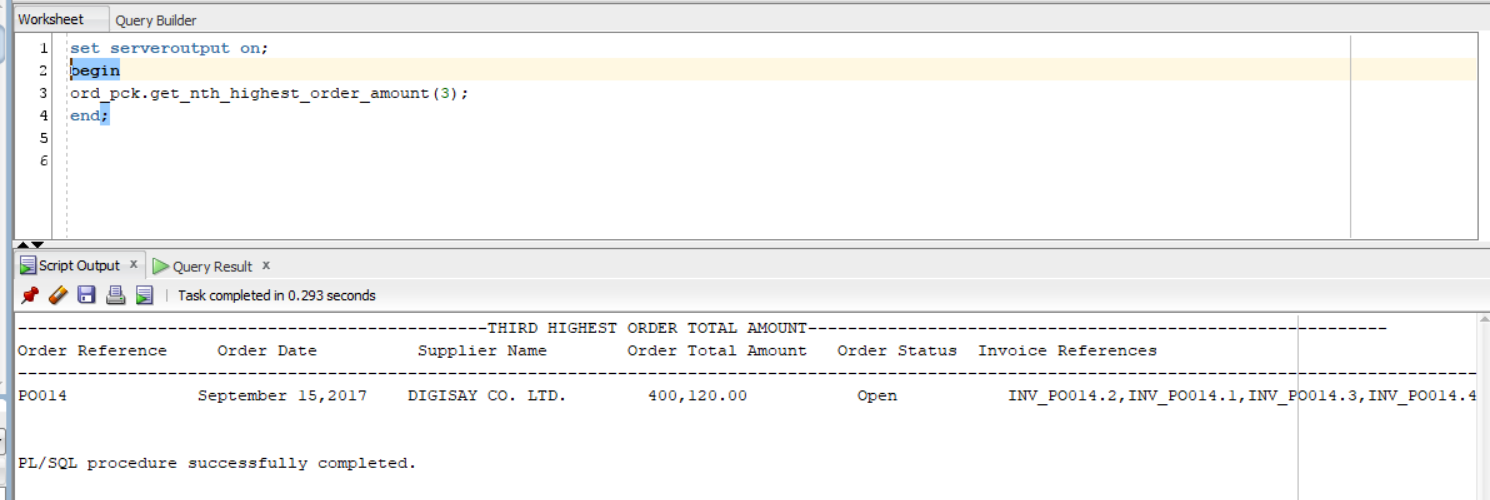
Answer:

set serveroutput on;

begin

ord\_pck.get\_nth\_highest\_order\_amount(3);

end;



6.List all suppliers with their respective ***number of orders and total amount ordered*** from them between the period of ***01 January 2017 and 31 August 2017***. Output details as per below. Implement a Stored Procedure or Function to return the required information.

|  |  |
| --- | --- |
| **Column Name** | **Specifications** |
| Supplier Name | As per record. |
| Supplier Contact Name | As per record. |
| Supplier Contact No. 1 | First contact number formatted as “999-9999” or “5999-9999”. |
| Supplier Contact No. 2 | Second contact number formatted as “999-9999” or “5999-9999”. |
| Total Orders | Total number of orders. |
| Order Total Amount | Format “99,999,990.00”. |

Answer:

set serveroutput on;

begin

ord\_pck.list\_suppliers\_with\_orders('01-JAN-2017','31-AUG-2017');

end;

