```
create or replace PACKAGE Pkg_FSS_Settlement AS
--*************************************************************
-- Pkg_FSS_Settlement: This package holds all the generic procedures/functions
--              that can used by any module.
-- There are also two sequences used for the program, they are:
--    RUNID_KEY: generated for creating primary key as RUNID for FSS_RUN_TABLE
--    SEQID_KEY: generated for creating part of primary key as LODGEMENTREF for
FSS_DAILY_SETTLEMENT
--
-- Developer  | When     | Comments
-- -----------|----------|-----------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--*************************************************************
--
   PROCEDURE DailyBankingSummary(p_date IN DATE default sysdate);
   PROCEDURE FraudReport;
   PROCEDURE DailySettlement;

END Pkg_FSS_Settlement;
/

create or replace PACKAGE BODY Pkg_FSS_Settlement AS
--*************************************************************
-- Pkg_FSS_Settlement: This package holds all the procedures/functions.
--              Some procedures/functions  can not be accessed from outside.
-- These procedures/functions are:
-- PROCEDURE SettleTransactions(p_newSettlement IN OUT NUMBER);
-- FUNCTION  Check_Amount_Date(p_amount IN NUMBER) RETURN boolean;
-- PROCEDURE DestBankFile;
-- PROCEDURE Update_RunTable(p_number IN NUMBER, p_outcome IN VARCHAR2, p_message IN
VARCHAR2);
--
-- Developer  | When     | Comments
-- -----------|----------|-----------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--*************************************************************
-------------------------------------------------------------------------------------------
----

--*************************************************************
-- Update_RunTable: This function takes a number and two
--                  strings to update FSS_RUN_TABLE in a logic way.
--
-- Parameters : IN
--              p_number : this is the runid passed from caller.
--              p_outcome: this is the runoutcome('SUCCESS','FAIL') passed from caller
--              p_outcome: this is the error message passed from caller
-- Developer  | When     | Comments
-- -----------|----------|-----------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--*************************************************************
--
PROCEDURE Update_RunTable(p_number IN NUMBER,
                          p_outcome IN VARCHAR2,
                          p_message IN VARCHAR2) IS
   --Allow commit without affecting the other transactions
   PRAGMA AUTONOMOUS_TRANSACTION;
   -- assign current runid in the current run of the program to v_run_id
   v_run_id Number := runid_key.CURRVAL;

BEGIN
```

```
--if p_number is provided, then insert a new reocrd in the run table using p_number as the
primary key-runid
IF p_number IS NOT NULL THEN
   INSERT
   INTO FSS_RUN_TABLE
   VALUES (p_number
          ,SYSDATE
          ,NULL
          ,NULL
          ,NULL);
ELSIF p_outcome='SUCCESS' THEN
   UPDATE FSS_RUN_TABLE
   SET RUNEND=sysdate,
       RUNOUTCOME='SUCCESS',
       REMARKS=p_message
       WHERE RUNID = v_run_id;
ELSIF p_outcome='FAIL' THEN
   UPDATE FSS_RUN_TABLE
   SET RUNEND=sysdate,
       RUNOUTCOME='FAIL',
       REMARKS=p_message
       WHERE RUNID = v_run_id;
END IF;

Commit;

EXCEPTION
   WHEN  OTHERS THEN
      common.log('Error occurs at Update_RunTable, error code ' || SQLERRM);
      Update_RunTable(NULL,'FAIL','Error occurs at updating the Run Table');
END Update_RunTable;

--************************************************************
-- Check_Amount_Date: This function takes a number to check whether a merchant's daily
settlement
--                    meets the minimum settlement amount considering the amount and the
current sysdate
--
-- Parameters : IN
--              p_amount : this is the amount passed from SettleTransactions procedure
--                         as the total settlement for a merchant.
--
-- Returns    : TRUE if the amount is greater than the minimum daily settlement or the
current day is the last day of the month,
--              else returns FALSE.
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
FUNCTION  Check_Amount_Date(p_amount IN NUMBER) RETURN boolean IS
   minimum_amount NUMBER;
BEGIN
--assign the value of the minimum daily settlement from FSS_REFERENCE to minimum_amount
SELECT REFERENCEVALUE*100 INTO minimum_amount FROM FSS_REFERENCE WHERE REFERENCEID='DMIN';
--if the current day is the last day of the month then return true
IF TRUNC(SYSDATE)=TRUNC(LAST_DAY(SYSDATE)) THEN
   RETURN TRUE;
ELSIF p_amount> minimum_amount THEN
   RETURN TRUE;
ELSE
   RETURN FALSE;
   END IF;
```

```
EXCEPTION
   WHEN OTHERS THEN
   Rollback;
   common.log('Error occurs at check_amount_date, error is ' || SQLERRM);
   Update_RunTable(NULL,'FAIL','Error occurs at check_amount_date') ;
END Check_Amount_Date;

--************************************************************
-- SettleTransaction: This function put new record in FSS_DAILY_SETTLEMENT table
--                    and update the FSS_DAILY_TRANSACTION table by stamping logementref
into it.
--
-- Parameters : IN OUT
--              p_newSettlement : this is the number of the total settlement which can be
passed back to the
--              DailySettlement procedure, so it can update the run table
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
PROCEDURE SettleTransactions(p_newSettlement IN OUT NUMBER) IS
   --set the lodgementref to be a date concatenated with a sequence number as the specific
requirement
   v_lodgement_id Number := to_char(sysdate, 'DDMMYYYY')||LPAD(seqid_key.NEXTVAL, 7, '0');
   --this cursor is used to select each merchant with their total settlement amount using
sum fuction and group by statement
   cursor c_merchant_Total is select f3.merchantid, sum(transactionamount) amount from
FSS_DAILY_TRANSACTION f1, FSS_TERMINAL f2, FSS_MERCHANT f3
   where f1.lodgementref IS NULL and f1.terminalid=f2.terminalid and
f2.merchantid=f3.merchantid group by f3.merchantid order by f3.merchantid ;
   r_merchant_total c_merchant_Total%ROWTYPE;

BEGIN

OPEN c_merchant_Total;
   LOOP
      FETCH c_merchant_Total INTO r_merchant_Total;
      EXIT WHEN c_merchant_Total%NOTFOUND;
      -- if the merchant's daily settlement is greater than the designed minimum
settlement, then insert a record into the FSS_DAILY_SETTLEMENT with v_lodgement_id as the
primary key
      IF CHECK_AMOUNT_DATE(r_merchant_Total.amount) THEN
         INSERT
         INTO
FSS_DAILY_SETTLEMENT(LODGEMENTREF,MERCHANTID,AMOUNT,SETTLEMENTDATE,PRINTSTATUS)
         VALUES (v_lodgement_id
                ,r_merchant_Total.MERCHANTID
                ,r_merchant_Total.amount
                ,sysdate
               ,'F');
         /*once a settlement record is added in FSS_DAILY_SETTLEMENT , then update the
FSS_DAILY_TRANSACTION by stamping lodgementref of settled settlement into
FSS_DAILY_TRANSACTION
         so that those transactions will only be settled once*/
         UPDATE FSS_DAILY_TRANSACTION
         SET LODGEMENTREF = v_lodgement_id
         WHERE TRANSACTIONNR IN(SELECT TRANSACTIONNR FROM FSS_DAILY_TRANSACTION F1,
FSS_TERMINAL F2, FSS_MERCHANT F3
                                WHERE F3.MERCHANTID=r_merchant_Total.MERCHANTID
                                AND F1.TERMINALID=F2.TERMINALID
                                AND F2.MERCHANTID=F3.MERCHANTID
```

```
                                AND F1.LODGEMENTREF IS NULL);
      -- set up next lodgementref
      v_lodgement_id:= to_char(sysdate, 'DDMMYYYY')||lpad(seqid_key.NEXTVAL, 7, '0');
      p_newSettlement:=p_newSettlement+1;
      END IF;
   END LOOP;
close c_merchant_Total;

EXCEPTION
   WHEN  OTHERS THEN
      Rollback;
      common.log('Error occurs at SettleTransactions, error is ' || SQLERRM);
      Update_RunTable(NULL,'FAIL','Error occurs at SettleTransactions');
END SettleTransactions;

--************************************************************
-- f_centre: This function return a text that's in the center of a line.
--
-- Parameters : IN
--              p_text : this is the text needs to be centered in a line.
--              p_pageWidth: This is the width of a page.
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
FUNCTION f_centre(p_text VARCHAR2,
                  p_pageWidth NUMBER) RETURN VARCHAR2 IS
--
   v_textWidth NUMBER;
BEGIN
   v_textWidth := LENGTH(p_text) / 2;
   RETURN LPAD(p_text, (p_pageWidth/2) + v_textWidth, ' ');
END;

--************************************************************
-- get_codes_value: This function return a value from PARAMETER table
--
-- Parameters : p_kind: the parameter kind
--              p_code: the destination for the kind
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
FUNCTION get_codes_value(p_kind VARCHAR2,
                         p_code VARCHAR2) RETURN VARCHAR2 is
--
v_value   PARAMETER.value%TYPE;
v_proc_name  VARCHAR2(50) := 'get_codes_value';
--
BEGIN
   select value INTO v_value
   from PARAMETER
   where kind = p_kind
   and   code = p_code;
   --
   RETURN v_value;
   --
EXCEPTION
   WHEN OTHERS THEN
```

```
       common.log(v_proc_name, 'Exception in get_codes_value with '||SQLERRM);
        RETURN null;
END;

--************************************************************
-- send_email: This procedure sends a Email with an attachment of Daily Settlement Report
to a nominated person
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
PROCEDURE Send_Email IS
--
   TYPE fileData is RECORD (
        fileData    CLOB);

   TYPE fileDataArray is TABLE of fileData
        INDEX BY BINARY_INTEGER;
   v_index_base CONSTANT NUMBER := 1;
   v_counter NUMBER:=v_index_base;
--
   v_file     utl_file.file_type;
   v_utlDir VARCHAR2(35) := 'XL_DIR';
   v_buffer   VARCHAR2(255);
--
   p_subject  VARCHAR2(50) := 'Daily Settlement Report on '||sysdate;
   p_message  VARCHAR2(50) := 'Please find the attached report below';
   p_recipient VARCHAR2(255);
   p_sender    VARCHAR2(50) := ' FinancialSettlementSystem@uts.edu.au';
   v_mailhost VARCHAR2(50) := 'postoffice.uts.edu.au';
   mail_conn       UTL_SMTP.connection;
   con_nl VARCHAR2(2) := CHR(13)||CHR(10);
   con_email_footer VARCHAR2(250) := 'This is an automatically generated email from the FSS
so please do not respond';
   v_boundary_text VARCHAR2(25) := 'Xudong Liu';
--
   report fileDataArray;

BEGIN
--
v_file := utl_file.fopen (v_utlDir,'DBS_'||to_char(sysdate,'DDMMYYYY')||'_XL'||'.txt',
'R');
--
LOOP
    BEGIN
      utl_file.get_line(v_file, v_buffer);
      report(v_counter).fileData:=v_buffer;
      v_counter:=v_counter+1;
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        EXIT;
    END;
END LOOP;
--
utl_file.fclose(v_file);
--
p_recipient:=get_codes_value('EMAIL_ADDRESS','ASS2_RECIPIENT');

mail_conn := UTL_SMTP.open_connection (v_mailhost, 25);
UTL_SMTP.helo (mail_conn, v_mailhost);
UTL_SMTP.mail (mail_conn, p_sender);
UTL_SMTP.rcpt (mail_conn, p_recipient);
UTL_SMTP.OPEN_DATA(mail_conn);
```

```
--
UTL_SMTP.WRITE_DATA(mail_conn,'From' || ':' || p_sender|| con_nl);
UTL_SMTP.WRITE_DATA(mail_conn, 'To: '||p_recipient||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn, 'Subject: '||p_subject||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Mime-Version: 1.0'||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Content-Type: multipart/mixed;
boundary="'||v_boundary_text||'"'||con_nl||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'--'||v_boundary_text||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Content-type: text/plain; charset=us-ascii'||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,con_nl||'Sent From Financial Settlement System(FSS)
'||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Please find the attached report below'||con_nl||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn, 'Regards'||con_nl||'FSS'||con_nl||con_nl);
UTL_SMTP.write_data (mail_conn, con_nl || con_email_footer||con_nl||con_nl);
--
UTL_SMTP.WRITE_DATA(mail_conn,con_nl||'--'||v_boundary_text||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Content-Type: application/octet-stream;
name="'||'DBS_29052015_XL.txt'||'"'||con_nl);
UTL_SMTP.WRITE_DATA(mail_conn,'Content-Transfer-Encoding: 7bit'||con_nl||con_nl);    --7bit
--
FOR i IN 1..report.COUNT LOOP
   UTL_SMTP.WRITE_DATA(mail_conn, report(i).fileData);
   UTL_SMTP.WRITE_DATA(mail_conn, chr(10));
END LOOP;
--
UTL_SMTP.WRITE_DATA(mail_conn,con_nl||'--'||v_boundary_text||'--'||con_nl);
UTL_SMTP.CLOSE_DATA(mail_conn);
UTL_SMTP.QUIT(mail_conn);

EXCEPTION
   WHEN utl_file.invalid_operation THEN
      utl_file.fclose(v_file);
      common.log('Unable to read Daily Settlement Report at DestBankSummary, error is ' ||
SQLERRM);
      Update_RunTable(NULL,'FAIL','Unable to read Daily Settlement Report') ;
   WHEN OTHERS THEN
      UTL_SMTP.CLOSE_DATA(mail_conn);
      common.log('Unable to open mail connection at Sent_Email, error is ' || SQLERRM);
      Update_RunTable(NULL,'FAIL','Unable to open mail connection at Sent_Email') ;
END send_email;

--************************************************************
-- DestBankFile: This procedure is used to print the daily deskbank file
--
-- Developer  | When     | Comments
-- -----------|----------|-----------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
PROCEDURE DestBankFile IS
   v_filePointer utl_file.file_type;
   v_utlDir VARCHAR2(35) := 'XL_DIR';
   v_utlFileName VARCHAR2(35);
   v_pageNr NUMBER := 1;
   v_pageWidth NUMBER := 120;
   v_organisationName Varchar2(50);
   v_organisationAccount Varchar2(16);
   --to cout the total settlement
   v_count number:=0;
   v_date Varchar2(6):=to_char(sysdate,'DDMMYY');
   v_total_credit number:=0;
   /*this cursor is used to select settlement record whose settlement date is the current
day and printstatus is 'F' which means it has not been printed yet.
   I can pass a collection of type from SettleTransactions to this procedure, making it
```

```
only print settlements once, which printstatus can do the same job,
   but considering the department might need to print deskbank file for a given day it is
better to have a column called printstatus to control the printing which can be ignored
when needed */
   cursor c_merchant_details is select f2.merchantid m_id, merchantaccounttitle
account_title, substr(merchantbankbsb,1,3)||'-
'||substr(merchantbankbsb,4,3)||merchantbankaccnr account_number, amount credit,
lodgementref,printstatus
   from fss_merchant f1, fss_daily_settlement f2 where printstatus='F' and
f1.merchantid=f2.merchantid and trunc(settlementdate)=trunc(sysdate);
   r_merchant_details c_merchant_details%ROWTYPE;

BEGIN

SELECT ORGACCOUNTTITLE,substr(ORGBSBNR,1,3)||'-'||substr(ORGBSBNR,4,3)||ORGBANKACCOUNT into
v_organisationName, v_organisationAccount from FSS_ORGANISATION;

SELECT 'DS_'||to_char(sysdate,'DDMMYYYY')||'_XL'||'.dat' INTO v_utlFileName FROM dual;

v_filePointer := utl_file.fopen(v_utlDir, v_utlFileName, 'W');
utl_file.put_line(v_filePointer,RPAD('0',18)||'01'||'WBC'||LPAD('S/CARD BUS
PAYMENTS',26)||LPAD('038559',13)||'INVOICES'||LPAD(v_date,10));

OPEN c_merchant_details;
   LOOP
      FETCH c_merchant_details into r_merchant_details;
      EXIT WHEN c_merchant_details%NOTFOUND;
      utl_file.put_line(v_filePointer,'1'||r_merchant_details.account_number||'
'||'50'||LPAD(r_merchant_details.credit,10,'0')||RPAD(r_merchant_details.account_title,33)|
|'F '||r_merchant_details.lodgementref||'032-797   001006'||'SMARTCARD TRANS
'||RPAD('0',8,'0') );
      --update printstatus to 'T' as True which means it has been processed and it should
be be printed out again
      UPDATE FSS_DAILY_SETTLEMENT
         SET PRINTSTATUS = 'T'
         WHERE LODGEMENTREF=r_merchant_details.LODGEMENTREF;

      v_total_credit:=v_total_credit+r_merchant_details.credit;
      v_count:=v_count+1;
END LOOP;
close c_merchant_details;
utl_file.put_line(v_filePointer,'1'||v_organisationAccount||'
'||'13'||LPAD(v_total_credit,10,'0')||RPAD(v_organisationName,33)||'N
'||'800500000000000'||'032-797   001006'||'SMARTCARD TRANS '||RPAD('0',8,'0') );
v_count:=v_count+1;
utl_file.put_line(v_filePointer,'7'||RPAD('999-
999',19)||RPAD('0',10,'0')||LPAD(v_total_credit,10,'0')||LPAD(v_total_credit,10,'0')||LPAD(
' ',20)||LPAD(v_count,6,'0'));
utl_file.fclose(v_filePointer);

EXCEPTION
   WHEN  OTHERS THEN
      utl_file.fclose(v_filePointer);
      common.log('Unable to write Daily Banking Summary at DestBankSummary, error is ' ||
SQLERRM);
      Update_RunTable(NULL,'FAIL','Unable to write Daily Banking Summary') ;
END DestBankFile;

--************************************************************
-- DailySettlement: This procedure is the main procedure
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
```

```
--************************************************************
--
PROCEDURE DailySettlement IS
   v_count_run number:=0;
   v_count_running number:=0;
   --v_count_newSettlement is used in updating run table so that it can display how many
new settlement has been settled.
   v_count_newSettlement Number:=0;
   is_running exception;
   already_run exception;
   --set up a runid for run table
   v_run_id Number := runid_key.NEXTVAL;
BEGIN
--insert a record into the run table when the program starts
Update_RunTable(v_run_id, NULL, NULL);
-- this statement is used to determine if the program has been run before.
SELECT count(*) into v_count_run from FSS_RUN_TABLE where runoutcome='SUCCESS' and
trunc(runend)=trunc(sysdate);
-- this statement is used to determine if the program is still running.
SELECT count(*) into v_count_running from FSS_RUN_TABLE where runstart is not null and
runend is null and trunc(runend)=trunc(sysdate);

IF v_count_run<>0 THEN
   raise already_run;
ELSIF v_count_running<>0 Then
   raise is_running;
ELSE
   INSERT
   INTO
FSS_DAILY_TRANSACTION(TRANSACTIONNR,DOWNLOADDATE,TERMINALID,CARDID,TRANSACTIONDATE,CARDOLDV
ALUE,TRANSACTIONAMOUNT,CARDNEWVALUE,TRANSACTIONSTATUS,ERRORCODE,LODGEMENTREF)
   SELECT
TRANSACTIONNR,DOWNLOADDATE,TERMINALID,CARDID,TRANSACTIONDATE,CARDOLDVALUE,TRANSACTIONAMOUNT
,CARDNEWVALUE,TRANSACTIONSTATUS,ERRORCODE,NULL FROM FSS_TRANSACTIONS
   MINUS
   SELECT
TRANSACTIONNR,DOWNLOADDATE,TERMINALID,CARDID,TRANSACTIONDATE,CARDOLDVALUE,TRANSACTIONAMOUNT
,CARDNEWVALUE,TRANSACTIONSTATUS,ERRORCODE,NULL FROM FSS_Daily_TRANSACTION;
END IF;
--v_count_newSettlement is used in updating run table so that it can display how many new
settlement has been settled .
SettleTransactions(v_count_newSettlement);
DestBankFile;
DailyBankingSummary;
Send_Email;
FraudReport;

-- update the run table as the program run successfully
Update_RunTable(NULL,'SUCCESS', 'The total number of new settlement is
'||v_count_newSettlement);
-- make changes permanent
COMMIT;

EXCEPTION
   WHEN is_running THEN
      common.log('Program is still running. Aborting this run');
      Update_RunTable(NULL,'FAIL','Someone tried to run the program while it was still
running.');
   WHEN already_run THEN
      common.log('Program already ran today');
      Update_RunTable(NULL,'FAIL','Program already ran today');
   WHEN  OTHERS THEN
      Rollback;
      common.log('An error occurs at DailySettlement,The error is ' || SQLERRM);
      Update_RunTable(NULL,'FAIL','An error occurs at DailySettlement');
```

```
END DailySettlement;

--************************************************************
-- DailyBankingSummary: This procedure print daily banking summary.
-- Parameters : IN
--              p_date : users can input a specific date to only print settlement records
which settled on the input date
--                       if the date is not given by users, the caller procedure will pass
sysdate to p_date
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
PROCEDURE DailyBankingSummary(p_date IN DATE default sysdate) IS
   v_filePointer utl_file.file_type;
   v_utlDir VARCHAR2(35) := 'XL_DIR';
   v_utlFileName VARCHAR2(35);
   v_pageNr NUMBER := 1;
   v_pageWidth NUMBER := 95;
   v_organisationName Varchar2(50);
   v_organisationAccount Varchar2(16);
   v_totalCredit number:=0;
   v_date Varchar2(11):=to_char(p_date, 'DD-Mon-YYYY');
   --this cursor is used to select each merchant's details from fss_daily_settlement table
joined by fss_merchant table
   cursor c_merchant_details is select f2.merchantid m_id, merchantaccounttitle m_title,
substr(merchantbankbsb,1,3)||'-'||substr(merchantbankbsb,4,3)||merchantbankaccnr
account_number, amount/100 credit
   from fss_merchant f1, fss_daily_settlement f2 where f1.merchantid=f2.merchantid and
trunc(settlementdate)=trunc(p_date);
   r_merchant_details c_merchant_details%ROWTYPE;

BEGIN

SELECT 'DBS_'||to_char(sysdate,'DDMMYYYY')||'_XL'||'.txt' INTO v_utlFileName FROM dual;

v_filePointer := utl_file.fopen(v_utlDir, v_utlFileName, 'W');

utl_file.put_line(v_filePointer, f_centre('SMARTCARD SETTLEMENT SYSTEM',v_pageWidth));
utl_file.put_line(v_filePointer, f_centre('DAILY DESKBANK SUMMARY',v_pageWidth));
utl_file.put_line(v_filePointer, 'Date '||v_date||LPAD('Page '||v_pageNr, 77));
utl_file.new_line(v_filePointer);
utl_file.put_line(v_filePointer, 'Merchant ID '||LPAD('Merchant Name',23)||LPAD('Account
Number', 30)||LPAD('Debit ', 14)||LPAD('Credit ', 13));
utl_file.put_line(v_filePointer,LPAD('-  ', 13,'-')||LPAD('-  ', 35,'-')||LPAD('-', 21,'-
')||RPAD(' ',2)||LPAD('-   ', 13,'-')||LPAD('-', 9,'-'));

OPEN c_merchant_details;
   LOOP
      FETCH c_merchant_details into r_merchant_details;
      EXIT WHEN c_merchant_details%NOTFOUND;
      utl_file.put_line(v_filePointer, r_merchant_details.m_id||RPAD('
',6)||RPAD(r_merchant_details.m_title,35)||RPAD(r_merchant_details.account_number,31)||LPAD
(to_char(r_merchant_details.credit,'999999.00'),12));
      v_totalCredit:=v_totalCredit+r_merchant_details.credit;
   END LOOP;
CLOSE c_merchant_details;

select ORGACCOUNTTITLE,substr(ORGBSBNR,1,3)||'-'||substr(ORGBSBNR,4,3)||ORGBANKACCOUNT into
v_organisationName, v_organisationAccount from FSS_ORGANISATION;
utl_file.put_line(v_filePointer, LPAD('
```

```
',15)||RPAD(v_organisationName,35)||RPAD(v_organisationAccount,20)||to_char(v_totalCredit,'
999999.00'));
utl_file.put_line(v_filePointer,LPAD(' ',69)||RPAD('-',13,'-')||' '||LPAD('-',10,'-'));
utl_file.put_line(v_filePointer,RPAD('BALANCE
TOTAL',70)||RPAD(to_char(v_totalCredit,'999999.00'),11)||LPAD(to_char(v_totalCredit,'999999
.00'),12));
utl_file.new_line(v_filePointer);
utl_file.new_line(v_filePointer);
utl_file.put_line(v_filePointer, 'Deskbank file Name : '||v_utlFileName);
utl_file.put_line(v_filePointer, RPAD('Dispatch Date',19)||': '||v_date);
utl_file.put_line(v_filePointer, f_centre('*** End of Report ***',v_pageWidth));
utl_file.fclose(v_filePointer);

EXCEPTION
   WHEN  OTHERS THEN
      utl_file.fclose(v_filePointer);
      common.log('Unable to write Daily Banking Summary Report at DailyBankingSummary,
error is ' || SQLERRM);
      Update_RunTable(NULL,'FAIL','Unable to write Daily Banking Summary Report at
DailyBankingSummary') ;
END DailyBankingSummary;

--************************************************************
-- FraudReport: This procedure print fraud reports.
--
-- Developer  | When     | Comments
-- -----------|----------|---------------------------------
-- Xudong Liu | 22/05/15 | Initial Creation
--
--************************************************************
--
PROCEDURE FraudReport IS
   --a type is defined regarding suspected fraud transaction's details, so it can be used
in a collection
   TYPE t_card_rec IS RECORD
       (transactionnr    NUMBER,
        terminalid       VARCHAR2(10),
        cardid           VARCHAR2(17),
        transactiondate  DATE,
        cardoldvalue     NUMBER,
        cardnewvalue     NUMBER);
   --a collection is defined to save suspected transaction's details
   TYPE t_transaction_array is TABLE OF t_card_rec INDEX BY BINARY_INTEGER;
   v_transaction_list   t_transaction_array;
   -- this cursor is used to select each cardid from FSS_DAILY_TRANSACTION
   Cursor c_cardid is select cardid from FSS_DAILY_TRANSACTION group by cardid order by
cardid;
   -- this cursor in cursor is used to select all transactions of a card when a cardid
number is given, order by transactionnr
   Cursor c_suspect_transaction(p_cardid varchar2) is select
transactionnr,terminalid,cardid,transactiondate, cardoldvalue,cardnewvalue from
FSS_DAILY_TRANSACTION where cardid=p_cardid order by transactionnr;
   v_old_value   number;
   v_new_value   number;
   v_index_base CONSTANT NUMBER := 1;
   --v_counter is used to determine the collection's position
   v_counter NUMBER:=v_index_base;
   --v_count is used to control the loop when printing the fraud transactions
   v_count NUMBER:=1;

   v_filePointer utl_file.file_type;
   v_utlDir VARCHAR2(35) := 'XL_DIR';
   v_utlFileName VARCHAR2(35);
   v_pageNr NUMBER := 1;
   v_pageWidth NUMBER := 100;
```

```
    v_date Varchar2(11):=to_char(sysdate, 'DD-Mon-YYYY');

 Begin
 /*cursor in cursor is used to get fraud transaction the basic idea is order each card's all
 transactions by transactiondate, one cursor is used to get each cardid
 and the other cursor uses this cardid as a parameter to select all transactions that belong
 to this card.Then it will compare the difference between the old value of the new
 transaction and new value of previous transaction  */
 FOR r_cardid IN c_cardid LOOP
    FOR r__suspect_transaction IN c_suspect_transaction(r_cardid.cardid) LOOP
    --bypass the first transaction of a card, only start with second transaction
    IF c_suspect_transaction%ROWCOUNT<>1
    THEN
       v_old_value:=r__suspect_transaction.cardoldvalue;
       IF v_old_value <= v_new_value
       Then
          v_new_value:=r__suspect_transaction.cardnewvalue;
       ELSE
          v_transaction_list(v_counter):=r__suspect_transaction;
          v_counter := v_counter+1;
          v_new_value:=r__suspect_transaction.cardnewvalue;
       END IF;
    ELSE
       v_new_value:=r__suspect_transaction.cardnewvalue;
    END IF;
    END LOOP;
 END LOOP;

 SELECT 'Fraud_'||to_char(sysdate,'DDMMYYYY')||'_XL'||'.txt' INTO v_utlFileName FROM dual;
 v_filePointer := utl_file.fopen(v_utlDir, v_utlFileName, 'W');
 utl_file.put_line(v_filePointer, f_centre('Fraud Report',v_pageWidth));
 utl_file.put_line(v_filePointer, 'Date '||v_date||LPAD('Page '||v_pageNr, 83));
 utl_file.new_line(v_filePointer);
 utl_file.put_line(v_filePointer, 'Transactionnr'||LPAD('Terminal ID',18)||LPAD('Card
 ID',18)||LPAD('Transaction Date',26)||LPAD('Old Value', 12)||LPAD('New Value', 12));
 utl_file.put_line(v_filePointer,LPAD('-', 13,'-')||RPAD(' ',7)||LPAD('-', 11,'-')||RPAD('
 ',6)||LPAD('-', 17,'-')||RPAD(' ',5)||LPAD('-', 16,'-')||'   '||RPAD('-',9,'-')||'
 '||LPAD('-', 9,'-')) ;
 --print all fraud transactions
 LOOP
    EXIT WHEN v_count> v_transaction_list.count;
    utl_file.put_line(v_filePointer, v_transaction_list(v_count).transactionnr||RPAD('
 ',15)||v_transaction_list(v_count).terminalid||RPAD('
 ',7)||v_transaction_list(v_count).cardid||RPAD(' ',8)
    ||v_transaction_list(v_count).transactiondate||RPAD('
 ',7)||LPAD(to_char(v_transaction_list(v_count).cardoldvalue/100,'999999.00'),9)||RPAD('
 ',2)||LPAD(to_char(v_transaction_list(v_count).cardnewvalue/100,'999999.00'),10));
    v_count:=v_count+1;
 END LOOP;
 utl_file.new_line(v_filePointer);
 utl_file.new_line(v_filePointer);
 utl_file.put_line(v_filePointer, 'Fraud Report file Name : '||v_utlFileName);
 utl_file.put_line(v_filePointer, RPAD('Dispatch Date',19)||': '||v_date);
 utl_file.put_line(v_filePointer, f_centre('*** End of Report ***',v_pageWidth));
 utl_file.fclose(v_filePointer);

 EXCEPTION
    WHEN  OTHERS THEN
       utl_file.fclose(v_filePointer);
       common.log('Unable to write Fraud Report, error is '|| SQLERRM);
       Update_RunTable(NULL,'FAIL','Unable to write Fraud Report');
 END FraudReport;

 END Pkg_FSS_Settlement;
```