COSC 344 ASSIGNMENT 4 - Matthew Brooker - 541670

PROGRAM 1

The first program is called **findRevenue**, which is written in C. This program allows the user to find the number of products that were purchased in the supermarket and its total revenue within a date range. This program outputs only one row.

To compile:

> Type **make findRevenue exe=findRevenue** at the terminal window after navigating to the appropriate directory.

Then to run:

> Type **./findRevenue** at the terminal window.

The user will be then prompted to enter a ‘beginning’ date. This is the start of the date range the user wants to check purchases for.

The date must be in this format:

> **‘DD-MM-YYYY’**

So typing something like:

> **04-05-2013**

or

> **21-7-2001**

will work fine.

The user will next be prompted to enter an ‘ending’ date. This is the end of the date range the user wants to check purchases for.

The program will work providing the user types in a date later than the ‘beginning’ one.

PROGRAM 2

This program is called **ProductInfo**, and is written in Java. There is another class called **Product** located in a file called **Product.java** that is used by **ProductInfo**.
This program outputs product details purchased within a date range. These details include productID, product name, price, department number with the SSN and first name of the customer who purchased them. This program can output multiple tuples (depending on the given dates).

To compile:

> Type **javac ProductInfo.java** at the terminal window after navigating to the appropriate directory.

Then to run:

> Type **java ProductInfo** at the terminal window.

The user will be then prompted to enter a beginning and end date on separate lines.

The dates must be in this format:

> **'DD-MM-YYYY'**

So typing something like:

> **04-05-2013**
> **07-09-2014**

or

> **21-7-2001**
> **23-08-2014**

will work fine.

The program will work providing the user types in a second date that is later than the first one.

findRevenue.pc

```
/* findRevenue.pc
 * Reads pass.dat and connects to Oracle.
 *
 * Outputs the total number of products
 * purchased and total revenue given a
 * date range.
 *
 * Matthew Brooker, 541670.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sqlca.h>
#include "getresponse.c"

/* Constant definitions */

#define USER_LEN   20
#define PWD_LEN    20
#define DATE_LEN   12

/* Return codes for SQL */

#define SUCCESS    0
#define NOT_LOGGED_IN -1017
#define NOT_FOUND          1403

/* Define host variables */

EXEC SQL BEGIN DECLARE SECTION;

  varchar  username[USER_LEN];
  varchar  password[PWD_LEN];
  int      h_sumquantity;
  double   h_revenue;
  char   h_date1[DATE_LEN];
  char   h_date2[DATE_LEN];

EXEC SQL END DECLARE SECTION;
```

```
/* Function prototypes */

void connect_to_oracle (void);
void sql_error (char *msg);

int main() {

  /* Connect to Oracle */

  connect_to_oracle();

  /* Prompt for a two dates and puts them
  *  into host variables.
  */

  printf("\nEnter the beginning date: ");
  scanf("%s", h_date1);

  printf("\nEnter the ending date: ");
  scanf("%s", h_date2);

  /* Get data from ORACLE */

  EXEC SQL
          SELECT SUM(c.quantity),SUM(c.quantity * p.price)
          INTO :h_sumquantity, :h_revenue
          FROM containing c, product p, shopping_list s
          WHERE c.productid = p.productid
          AND s.shopping_date = c.shopping_date
          AND c.shopping_date BETWEEN TO_DATE(:h_date1, 'DD-MM-YYYY')
          AND TO_DATE(:h_date2, 'DD-MM-YYYY');


  /* Print a heading and the data */

  printf("\nPRODUCTS SOLD          TOTAL REVENUE\n");
  printf("-------------       -------------\n");
  printf("%-17d %-10.2f\n", h_sumquantity, h_revenue);

  /* Disconnect from ORACLE */

  EXEC SQL COMMIT WORK RELEASE;
```

```
  return(0);
}

void connect_to_oracle (void) {

  FILE *passfile;

  /* Open pass.dat.  If not successful, print
   * an error messge and exit.
   */

  if (0 == (passfile = fopen("pass.dat", "r"))) {
          printf("Cannot open pass.dat\n");
          printf("Program exiting\n");
          exit(-1);
  }

  /* Read the data from the file
   * and terminate the varchar strings.
   */

  getresponse((char *)username.arr, sizeof(username.arr), passfile);
  username.len = strlen((char *) username.arr);
  getresponse((char *)password.arr, sizeof(password.arr), passfile);
  password.len = strlen((char *) password.arr);

  /* Close the file. */

  fclose(passfile);

  printf("\nConnecting to ORACLE\n");

  EXEC SQL CONNECT :username IDENTIFIED BY :password;

  if (NOT_LOGGED_IN == sqlca.sqlcode) {
          printf("  Not connected\n");
          exit(-1);
  } else if (SUCCESS == sqlca.sqlcode) {
          printf("  Connected to ORACLE\n");
  } else {
          sql_error("Error logging into Oracle");
  }
```

```
  return;
}

void sql_error (char *msg) {

  char err_msg[200];
  size_t buf_len, msg_len;

  printf("\n%s\n", msg);
  buf_len = sizeof(err_msg);
  sqlglm(err_msg, &buf_len, &msg_len);
  printf("%.*s\n", msg_len, err_msg);
  EXEC SQL ROLLBACK RELEASE;
  exit(1);
}
```

getresponse.c

```
/* GETRESPONSE - This is a function for safely reading input
                 from a file or the keyboard.

   buffer -  is a character array for storing the data read
   limit -   is the size of the buffer.  WARNING: This number
          must not exceed the size of the buffer declared
          in the calling function.
   *whence - is where to read data from.  It can be an
          opened file pointer or could be stdin
*/

int getresponse (char buffer[], int limit, FILE *whence) {
  int c, i =0 ;
  /* While we have not reached the end of the file or the
          end of a line: get a character; decrement limit;
          if limit is greater than zero, put the character into
          the buffer; if limit does hit zero, output a message
          that we are truncating the input. */
  while ( ((c = getc(whence)) != EOF) && (c != '\n') ) {
          if (--limit > 0) {
          buffer[i++] = c;
          }
          if (limit == 0) {
          fprintf(stderr, "Warning: input truncated to length %d\n", i);
          }
  }

  /* Add the string terminator and return the number of characters.*/
  buffer[i] = '\0';
  return i;
}
```

ProductInfo.java

```
/*
  File: ProductInfo.java
  September 2014
  Matthew Brooker 541670.
*/

import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * This program outputs the product details of all
 * the purchases within a given date range, and who
 * made them.
 *
 * @author Matthew Brooker
 */

public class ProductInfo {


        public static void main (String[] args) {
    new ProductInfo().go();
        }

        // This is the function that does all the work
        private void go() {

    // Read pass.dat
    UserPass login = new UserPass();
    String user = login.getUserName();
    String pass = login.getPassWord();
    String host = "silver";

    Connection con = null;
    try {
        // Register the driver and connect to Oracle
        DriverManager.registerDriver
         (new oracle.jdbc.driver.OracleDriver());
        String url = "jdbc:oracle:thin:@" + host + ":1527:cosc344";
        System.out.println("url: " + url);
```

```
        con = DriverManager.getConnection(url, user, pass);
        System.out.println("Connected to Oracle");

        /* This prepared statement outputs the details of all purchases
        within a given date range, and who made them.
        */
        PreparedStatement pstmt = con.prepareStatement("SELECT a.ssn, a.fname, a.lname,
p.productid, " +
"p.product_name, p.price, p.dno " +
"FROM product p, containing c, customer a, shopping_list s " +
"WHERE c.productid = p.productid " +
"AND s.shopping_date = c.shopping_date " +
"AND a.ssn = s.ssn " +
"AND c.shopping_date BETWEEN TO_DATE(?, 'DD-MM-YYYY') AND
TO_DATE(?,'DD-MM-YYYY')");


        System.out.println("Please enter two dates on separate lines to specify " +
                        "what time period you wish to check all product's " +
                        "purchased in this form: DD-MM-YYYY. Be sure to type " +
                        "the earlier date first.");

        Scanner input = new Scanner(System.in);

        if (input.hasNextLine()) {
        String stringDate1 = input.nextLine();
        pstmt.setString(1, stringDate1);
        }
        else {
        System.out.println("You didn't enter a date!");
        System.exit(1);
        }

        if (input.hasNextLine()){
        String stringDate2 = input.nextLine();
        pstmt.setString(2, stringDate2);

        }
        else {
        System.out.println("You didn't enter a second date!");
        System.exit(1);
        }
```

```
        ResultSet rslt = pstmt.executeQuery();

        // Calls print array which handles the rest of the product output.
        printArray(rslt);


 } catch (SQLException e) {
        System.out.println(e.getMessage());
        System.exit(1);

 } finally {
        if (con != null) {
         try {
                  con.close();
         } catch (SQLException e) {
                  quit(e.getMessage());
         }
        }
 }
        }  // end go()

/**
        * printArray executes the database extraction using the result set
        * query and puts each selected tuple in an object called Product, to
        * add to an array list of products which are finally printed in
        * appropriate formatting.
        *
        *  @param rslt.
        *
        */
private void printArray(ResultSet rslt) {

        // The array list of Product objects.
        ArrayList<Product> products = new ArrayList<Product>();

        /* While there is another tuple, store each selected attribute
        * value, create a product object with these attributes, and add
        * it to the arraylist for printing.
        */
        try {
        while (rslt.next()) {
```

```
        String ssn = rslt.getString("ssn");
        String fname = rslt.getString("fname");
        String lname = rslt.getString("lname");
        String productid = rslt.getString("productid");
        String product_name = rslt.getString("product_name");
        double price = rslt.getDouble("price");
        int dno = rslt.getInt("dno");

        Product p = new Product(ssn, fname, lname, productid, product_name, price, dno);
        products.add(p);

        }
        }
        catch (SQLException e) {
        System.out.println(e.getMessage());
        System.exit(1);
        }
        // For the output table heading...
        System.out.println("SSN       FNAME          LNAME          PROD_ID
PRODUCT_NAME" +
        "         PRICE DNO");
        System.out.println("---          -----      -----      -------    ------------" +
        "          -----      ---");

        /* Print out all products using the getters found in Product class,
        the attributes productid, product_name, price and dno */
        for (Product pr: products){
        System.out.format("%-11s %-8s %-11s %-11s %-20s %-8.2f %-8d\n",
                pr.getSsn(), pr.getFname(), pr.getLname(),
                pr.getId(), pr.getName(), pr.getPrice(), pr.getDno());
        }
  }

        // Used to output an error message and exit
        private void quit(String message) {
   System.err.println(message);
   System.exit(1);
        }

}
```

Product.java

```
/*
  File: Product.java
  September 2014
  Matthew Brooker 541670.
*/
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * Deals with Product table information relating to query.
 *
 * @author Matthew Brooker
 */

public class Product {

   // Class variables.
   String ssn;
   String fname;
   String lname;
   String productid;
   String product_name;
   double price;
   int dno;

   /**
         * This is the Product constructer that sets the values of objects
         *
         * @param ssn, fname, lname, productid, product_name, price, dno.
         * These parameters are all the sql values from the select statement.
         * They represent the attribute values printed row by row as a table.
         */

   public Product(String ssn, String fname, String lname, String productid,
                  String product_name, double price, int dno) {
         this.ssn = ssn;
         this.fname = fname;
         this.lname = lname;
         this.productid = productid;
         this.product_name = product_name;
```

```
        this.price = price;
        this.dno = dno;
    }

    /**
        * These are the getter methods that return the output values.
        *
        * @return ssn, fname, lname, productid, product_name, price, dno.
        */
    public String getSsn() {
        return ssn;
    }

    public String getFname() {
        return fname;
    }

    public String getLname() {
        return lname;
    }

    public String getId() {
        return productid;
    }

    public String getName() {
        return product_name;
    }

    public double getPrice() {
        return price;
    }

    public int getDno() {
        return dno;
    }
}
```

UserPass.java

```
/*
  File:  UserPass.java
  July 2002
*/

import java.io.*;
import java.util.*;
import java.lang.*;

/**
 * Reads a username and password from a file called pass.dat.
 *
 * @author Paul Werstein
 */

public class UserPass {

        private String password;
        private String username;

        // Constructor - Also reads the username and password
        //                  from the file.
        public UserPass () {
    String line = null;
    String passwordFile = "pass.dat";
    try {
        BufferedReader inFile =
         new BufferedReader(new FileReader(passwordFile));
        // Read the username from the file and store it.
        if ((line = inFile.readLine()) == null) {
         quit(passwordFile + " is empty");
        }
        StringTokenizer tok = new StringTokenizer(line);
        if (tok.countTokens() != 1) {
         quit("Username line has an error");
        }
        username = tok.nextToken();
        // Read the password from the file and store it.
        if ((line = inFile.readLine()) == null) {
         quit(passwordFile + " has a bad format");
        }
```

```
            tok = new StringTokenizer(line);
            if (tok.countTokens() != 1) {
             quit("Password line has an error");
            }
            password = tok.nextToken();
        } catch (FileNotFoundException e) {
            quit("The file, " + passwordFile + ", was not found.");
        } catch (IOException e) {
            quit("An error occured trying to read " + passwordFile);
        }
            }

            // Returns the password

            public String getPassWord() {
        return password;
            }

            // Returns the username

            public String getUserName() {
        return username;
            }

            // Used for printing reasons for exceptions or errors.

            private void quit(String message) {
        System.err.println(message);
        System.exit(1);
            }

    } // end class UserPass
```

Load.sql

```
DROP TABLE containing;
DROP TABLE product;
DROP TABLE shopping_list;
DROP TABLE customer;
DROP TABLE employee cascade constraints;
DROP TABLE department cascade constraints;




CREATE TABLE department
        (dnumber        INT     PRIMARY KEY,
        dname           VARCHAR2(15) NOT NULL UNIQUE,
        dcontact_number CHAR(9)  NOT NULL UNIQUE,
        mgrssn          CHAR(9)       NOT NULL,
        mgrstartdate    DATE);


INSERT INTO department VALUES
        (1,'Produce','112392348','123456789', TO_DATE('22-05-1988','DD-MM-YYYY'));
INSERT INTO department VALUES
        (2,'Butchery','124356779','987654321', TO_DATE('01-01-1995','DD-MM-YYYY'));
INSERT INTO department VALUES
        (3,'Grocery','138556110','888665555', TO_DATE('19-06-1981','DD-MM-YYYY'));
INSERT INTO department VALUES
        (4,'Chilled Foods','148224661','111100000', TO_DATE('31-12-2004','DD-MM-YYYY'));
INSERT INTO department VALUES
        (5,'Liquor','158545766','158345766', TO_DATE('31-12-2004','DD-MM-YYYY'));




CREATE TABLE employee
        (ssn    CHAR(9)       PRIMARY KEY,
        fname  VARCHAR2(10)  NOT NULL,
        minit   CHAR,
        lname  VARCHAR2(20)  NOT NULL,
        bdate   DATE,
        area  VARCHAR2(20),
        sex     CHAR,
        salary   NUMBER(6),
```

```
        hours   NUMBER(2),
        superssn CHAR(9)
        CONSTRAINT superssn_cnst REFERENCES employee(ssn) DISABLE,
        dno     INT             NOT NULL
        CONSTRAINT dno_cnst REFERENCES department(dnumber) DISABLE);

ALTER TABLE employee ENABLE CONSTRAINT dno_cnst;

        INSERT INTO employee VALUES
        ('123456789','Tim','L','Jones',TO_DATE('24-10-1992','DD-MM-YYYY'),
        'Mornington','M',45000, 40, NULL, 1);
        INSERT INTO employee VALUES
        ('987688888','Rose','F','Petersond',TO_DATE('05-05-1989','DD-MM-YYYY'),
        'Maori Hill','F', 10000, 15,'123456789', 1);
        INSERT INTO employee VALUES
        ('333445555','Earl','V','Vonstrozzenburger',TO_DATE('16-09-1977','DD-MM-YYYY'),
        'North Dunedin','M',35000, 40,'123456789', 1);
        INSERT INTO employee VALUES
        ('987654321','Pete','G','Mcgee',TO_DATE('20-09-1965','DD-MM-YYYY'),
        'South Dunedin','M',45000, 40, NULL,2);
        INSERT INTO employee VALUES
        ('223691415','Doug','M','Glatt',TO_DATE('03-04-1983','DD-MM-YYYY'),
        'North Dunedin','M',35000, 40, '987654321',2);
        INSERT INTO employee VALUES
        ('239715567','Katie','S','Margaret',TO_DATE('01-07-1995','DD-MM-YYYY'),
        'North Dunedin','F',13000, 20,'987654321', 2);
        INSERT INTO employee VALUES
        ('888665555','Tom','C','Johnson',TO_DATE('02-06-1991','DD-MM-YYYY'),
        'Roslyn','M',45000, 40, NULL,3);
        INSERT INTO employee VALUES
        ('303012889','Jessica','B','Stevens',TO_DATE('19-11-1998','DD-MM-YYYY'),
        'North East Valley','F', 9000, 12,'888665555', 3);
        INSERT INTO employee VALUES
        ('887722669','Sophie','R','Smith',TO_DATE('05-06-1991','DD-MM-YYYY'),
        'Pine Hill','F', 9000, 40,'888665555',3);
        INSERT INTO employee VALUES
        ('111100000','James','F','Marshall',TO_DATE('03-07-1994','DD-MM-YYYY'),
        'South Dunedin','M',45000, 40, NULL,4);
        INSERT INTO employee VALUES
        ('999887777','Alicia','J','Zelaya',TO_DATE('19-07-1968','DD-MM-YYYY'),
        'Caversham','M',25000, 40, '111100000',4);
        INSERT INTO employee VALUES
        ('666884444','Ramesh','K','Narayan',TO_DATE('15-09-1962','DD-MM-YYYY'),
```

```
        'Mornington','M', 35000, 40,'111100000',4);
        INSERT INTO employee VALUES
        ('158345766','Joyce','A','English',TO_DATE('31-07-1972','DD-MM-YYYY'),
        'Central Dunedin','M',35000,40,NULL,5);
        INSERT INTO employee VALUES
        ('453453453','Ahmad','V','Jabbar',TO_DATE('29-03-1969','DD-MM-YYYY'),
        'Central Dunedin','M',35000,40,'158345766',5);
        INSERT INTO employee VALUES
        ('992134455','James','E','Irnbru',TO_DATE('10-11-1937','DD-MM-YYYY'),
        'Central Dunedin','M',35000,40,'158345766',5);

ALTER TABLE employee ENABLE CONSTRAINT superssn_cnst;


CREATE TABLE customer
        (ssn            CHAR(9) PRIMARY KEY,
        fname           VARCHAR2(10) NOT NULL,
        minit           CHAR,
        lname           VARCHAR2(20) NOT NULL,
        ccontact_number CHAR(10) NOT NULL UNIQUE,
        area            VARCHAR2(20),
        sex             CHAR);


        INSERT INTO customer VALUES ('345876567','Rupert','P','Princeton','0271815678',
                'Roslyn','M');
        INSERT INTO customer VALUES ('119982732','John','R','Bert','0275642231',
                'North Dunedin','M');
        INSERT INTO customer VALUES ('453428890','Joanne','G','Rutherford','0224351677',
                'South Dunedin','F');
        INSERT INTO customer VALUES ('556674390','Sarah','L', 'Edwards', '0221556783',
                'Mornington', 'F');


CREATE TABLE shopping_list
        (shopping_date DATE  PRIMARY KEY,
        num_products  CHAR(12),
        ssn             CHAR(9)   NOT NULL REFERENCES customer(ssn) ON DELETE SET
NULL);

        INSERT INTO shopping_list VALUES
        (TO_DATE('16-05-2014, 5:34 P.M.', 'DD-MM-YYYY, HH:MI P.M.'),4,'345876567');
```

```
INSERT INTO shopping_list VALUES
(TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'),8,'345876567');

INSERT INTO shopping_list VALUES
(TO_DATE('14-06-2014, 7:10 A.M.', 'DD-MM-YYYY, HH:MI A.M.'),3,'119982732');

INSERT INTO shopping_list VALUES
(TO_DATE('2-07-2014, 12:39 P.M.', 'DD-MM-YYYY, HH:MI A.M.'),2,'453428890');

INSERT INTO shopping_list VALUES
(TO_DATE('22-08-2014, 3.35 P.M.', 'DD-MM-YYYY, HH:MI A.M.'),5,'556674390');



CREATE TABLE product
(productid      CHAR(4)       PRIMARY KEY,
product_name   VARCHAR(20)   NOT NULL,
price          NUMBER(8, 2),
dno            INT           NOT NULL REFERENCES department(dnumber) ON
DELETE CASCADE);

INSERT INTO product VALUES(1000, 'Flyspray', 7.50, 3);
INSERT INTO product VALUES(1001, 'Chocolate Cake', 5.30, 3);
INSERT INTO product VALUES(1002, 'Frozen Pizza', 8.50, 4);
INSERT INTO product VALUES(1003, '6 Pack Beer', 14.00, 5);
INSERT INTO product VALUES(1004, 'Avocado', 2.00, 1);
INSERT INTO product VALUES(1005, 'Stawberries', 6.99, 1);
INSERT INTO product VALUES(1006, 'Lettuce', 3.49, 1);
INSERT INTO product VALUES(1007, 'Carrots 1kg', 5.99, 1);
INSERT INTO product VALUES(1008, 'Chicken Breasts', 13.99, 2);
INSERT INTO product VALUES(1009, 'Rump Steak', 5.99, 2);
INSERT INTO product VALUES(1010, 'Pork Sausages 1kg', 9.99, 2);
INSERT INTO product VALUES(1011, 'Ice Cream', 3.29, 4);
INSERT INTO product VALUES(1012, 'Hash Browns', 8.50, 4);
INSERT INTO product VALUES(1013, 'Noodles', 3.50, 3);
INSERT INTO product VALUES(1014, 'Spagetti', 2.99, 3);
INSERT INTO product VALUES(1015, 'Beef Jerkey', 4.50, 3);
INSERT INTO product VALUES(1016, 'Tortillas', 4.99, 3);
INSERT INTO product VALUES(1017, 'Tomato Sauce', 3.49, 3);
INSERT INTO product VALUES(1018, 'Ajax Spray and Wipe', 4.00, 3);
INSERT INTO product VALUES(1019, 'Nutella', 5.30, 3);
INSERT INTO product VALUES(1020, 'Olive Oil', 4.49, 3);
INSERT INTO product VALUES(1021, 'Butter', 5.99, 4);
```

```
INSERT INTO product VALUES(1022, 'Yogurt', 4.99, 4);
INSERT INTO product VALUES(1023, 'Bread', 2.99, 3);
INSERT INTO product VALUES(1024, 'Cookies', 4.50, 3);
INSERT INTO product VALUES(1025, 'Doughnut', 3.50, 3);
INSERT INTO product VALUES(1026, 'Pasta Sauce', 3.00, 3);
INSERT INTO product VALUES(1027, 'Tea Towels', 2.89, 3);
INSERT INTO product VALUES(1028, 'Dish Cloth', 2.00, 3);
INSERT INTO product VALUES(1029, 'Container', 5.50, 3);
INSERT INTO product VALUES(1030, 'Coke 1.5l', 1.80, 3);



CREATE TABLE containing
        (shopping_date DATE          REFERENCES shopping_list(shopping_date) ON DELETE
CASCADE,
        productid       CHAR(4)  REFERENCES product(productid) ON DELETE CASCADE,
        quantity        INT     NOT NULL,
        PRIMARY KEY(shopping_date, productid));

        INSERT INTO containing VALUES
        (TO_DATE('16-05-2014, 5:34 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1016, 1);

        INSERT INTO containing VALUES
        (TO_DATE('16-05-2014, 5:34 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1008, 1);

        INSERT INTO containing VALUES
        (TO_DATE('16-05-2014, 5:34 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1004, 2);


        INSERT INTO containing VALUES
        (TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1000, 1);

        INSERT INTO containing VALUES
        (TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1007, 1);

        INSERT INTO containing VALUES
        (TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1017, 1);

        INSERT INTO containing VALUES
        (TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1018, 1);

        INSERT INTO containing VALUES
```

```
(TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1029, 2);

INSERT INTO containing VALUES
(TO_DATE('20-05-2014, 10:54 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1030, 2);


INSERT INTO containing VALUES
(TO_DATE('14-06-2014, 7:10 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1024, 1);

INSERT INTO containing VALUES
(TO_DATE('14-06-2014, 7:10 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1025, 1);

INSERT INTO containing VALUES
(TO_DATE('14-06-2014, 7:10 A.M.', 'DD-MM-YYYY, HH:MI A.M.'), 1030, 1);


INSERT INTO containing VALUES
(TO_DATE('2-07-2014, 12:39 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1012, 2);


INSERT INTO containing VALUES
(TO_DATE('22-08-2014, 3:35 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1019, 1);

INSERT INTO containing VALUES
(TO_DATE('22-08-2014, 3:35 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1027, 1);

INSERT INTO containing VALUES
(TO_DATE('22-08-2014, 3:35 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1011, 1);

INSERT INTO containing VALUES
(TO_DATE('22-08-2014, 3:35 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1006, 1);

INSERT INTO containing VALUES
(TO_DATE('22-08-2014, 3:35 P.M.', 'DD-MM-YYYY, HH:MI P.M.'), 1003, 1);



COMMIT;
```