Assignment 2 5%

Due Date: November 2, 2017 midnight

This assignment covers the material from Chapters 8 DS book and SQL DDL and DML statements.

Part 1 PL/SQL Procedure 3 %

The following relations (based on chapter 4, DS textbook) form part of a relational database schema:

Hotel	(hotelNo, hotelName, hotel_address, city, province, country)
Room	(roomID, roomNo, hotelNo, type, and price)
Booking	(BookingID, guestNo, dateFrom, dateTo, roomID)
Guest	(guestNo, guestName, guestAddress, comments)

Where Hotel contains hotel details and hotelNo is the primary key; Room contains room details for each hotel and roomID is the primary key; Room type specifies “single”, “double” or “family.” Booking contains details of the bookings and BookingID is the primary key; and the Guest table contains guest details and guestNo is the primary key. Comments are a string of characters and digits stored a variable character type with maximum 128 characters.

Preliminary task:

Create required tables (use **prefix A2_ for their names**) and add data. You can use CASE tool or SQLDeveloper.

1. **Problem:** Our reservation center gets a lot of "last minute" requests for a one night reservation starting the same day. The staff would like to have a quick method of finding an available room (any type any hotel) in a specific city. For example: a guest arrives to Kamloops in the late afternoon and needs to stay for one night in Kamloops. The system should find all hotels in Kamloops with at least one available room for tonight.

Hints for solution:

Create a stored procedure LIST_AVAILABLE with one input parameter: p_city. This procedure should list all hotels in the specified city with at least one room available for today's (SYSDATE) night. The room is available if there are no reservations for today's night. List the "title": Hotels available in city ….for 2016-10-26 and a line for each hotel name and address.

```
CREATE OR REPLACE PROCEDURE CHECK_AVAILABLE_ROOMS (P_CITY VARCHAR)
    IS
    hotel_number NUMBER;
    name_hotel VARCHAR(35);
    hotel_address VARCHAR(60);
    hotel_city VARCHAR(20);

    CURSOR rooms_cursor IS
        SELECT HOTELNO, HOTELNAME, HOTEL_ADDRESS, CITY FROM
            (SELECT H.HOTELNO, H.HOTELNAME, H.HOTEL_ADDRESS, H.CITY
                FROM A2_ROOM R LEFT OUTER JOIN A2_BOOKING B ON R.ROOMID =
B.ROOMID
                JOIN A2_HOTEL H ON R.HOTELNO = H.HOTELNO
                WHERE
                R.ROOMID NOT IN (SELECT ROOMID FROM A2_BOOKING)
                OR B.DATETO < SYSDATE
                GROUP BY H.HOTELNO, H.HOTELNAME, H.HOTEL_ADDRESS, H.CITY
                ORDER BY H.HOTELNO)
        WHERE UPPER (CITY) =UPPER (P_CITY);

    BEGIN

        OPEN rooms_cursor;
            LOOP
                FETCH rooms_cursor INTO
hotel_number,name_hotel,hotel_address,hotel_city;
                EXIT WHEN rooms_cursor%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE (hotel_number||'   '||name_hotel||'
'||hotel_address||'   '||UPPER (hotel_city));
            END LOOP;
        CLOSE rooms_cursor;
END;
```

```
EXECUTE CHECK_AVAILABLE_ROOMS ('KAMLOOPS');
1   HOTEL 1   KAMLOOPS   KAMLOOPS
7   HETEL 7   PG 2   KAMLOOPS
```

HETEL 5 is not included since all its rooms are booked.

2. **Problem:** The database designer has forgotten to create an attribute for the phone numbers. The hotel staff was using the comments (attribute) to enter textual comments and also phone numbers. Ooops.. The hotel manager is very upset and your task is to provide a temporarily solution. The manager wants to have a list of the guests (guestName, and possible phone number extracted from the comments. This has to be done within one hour (before the manager gets back from lunch).

   Hints for solution:

Create an SQL statement to list the guests (guest names) and the parts of the comments matching phone patterns. The examples of phone numbers: (250) 333 9999, 250-371-5592, 371-5592 250.333.4444. Good news: Oracle supports Regular Expressions (!). Idea: Use the regular expressions to find potential phone numbers in comments. Oracle starting from 10g uses regular expressions. Read Chapter 10 (Oracle 11g SQL book) pp. 357-359 (on reserve).Chapter 10 (Oracle 12c SQL book) pp. 374-377. There are five regexp functions in Oracle SQL and PL/SQL: REGEXP_LIKE, REGEXP_INSTR, REGEXP_REPLACE, REGEXP_SUBSTR, and REGEXP_COUNT.

Submit

- Listing of the Stored Procedure
- Test of the procedure showing the available rooms
- SQL query to find the phone numbers in comments.
- Test running the SQL query

```
SELECT GUESTNAME,REGEXP_SUBSTR(COMMENTS,'[0-9]{10}|[0-9]{3}.{1}[0-
9]{3}.{1}[0-9]{4}|.{1}[0-9]{3}.{1}.{1}[0-9]{3}.{1}[0-9]{4}|[+]{1}[0-
9]{2}[0-9]{3}[0-9]{3}[0-9]{4}|[+]{1}[0-9]{2}.{1}[0-9]{3}.{1}[0-9]{3}.{1}[0-
9]{4}|[0-9]{7}|[0-9]{3}.{1}[0-9]{4}')"Possible Phone Number" FROM A2_GUEST;
```

	GUESTNAME	Possible Phone Number
1	GUEST ONE	250-682-0867
2	GUEST TWO	2509999999
3	GUEST THREE	(250) 333 9999
4	GUEST FOUR	250.333.4444
5	GUEST FIVE	222-1234
6	GUEST SIX	123 456 7891
7	GUEST SEVEN	(250)-424-9999
8	GUEST EIGHT	+12 345 678 9999
9	GUEST NINE	+124445667787

```
SELECT GUESTNAME "Name",
    REGEXP_SUBSTR (COMMENTS,
    '([+]?[0-9]{2})?[(,[:space:]]{0,1}[0-9]{3}[),.,-,[:space:]]{0,2}[0-
9]{3,}[),.,-,[:space:]]{0,2}[0-9]{4,}')
    "Possible Phone Number"
    FROM A2_GUEST;
```

Name	Possible Phone Number
GUEST ONE	
GUEST TWO	2509999999
GUEST THREE	(250) 333 9999
GUEST FOUR	250.333.4444
GUEST FIVE	
GUEST SIX	123 456 7891
GUEST SEVEN	
GUEST EIGHT	+12 345 678 9999
GUEST NINE	+124445667787

- RETURNS MOST OF THE NUMBERS FROM THE COMMENTS

Part 2 Archiving 2 %

Bookings should be archived after 2 years. Create the ARCHIVED_BOOKING table which will have the same columns as the BOOKING table and additionally a column called ARCHIVED_DATE. Create a stored procedure to archive bookings older than 2 years (based on DATE_TO and SYSDATE). Add the old bookings to the ARCHIVED_BOOKING table and remove the old bookings from the BOOKING table.

Submit

1. Create statement for the ARCHIVED_BOOKING table
2. Listing of the Stored Procedure
3. Test demonstrating the execution of the archived procedure

```
CREATE TABLE A2_ARCHIVED_BOOKING
    AS (SELECT * FROM A2_BOOKING WHERE 0=1);
ALTER TABLE A2_ARCHIVED_BOOKING
    ADD (ARCHIVED_DATE DATE DEFAULT SYSDATE NOT NULL);


CREATE OR REPLACE PROCEDURE ARCHIVE_OLD_BOOKINGS
    AS

BEGIN
    INSERT INTO A2_ARCHIVED_BOOKING (BOOKINGID, GUESTNO, DATEFROM, DATETO,
ROOMID, ARCHIVED_DATE)
        SELECT A.BOOKINGID, A.GUESTNO, A.DATEFROM, A.DATETO, A.ROOMID,
SYSDATE
        FROM A2_BOOKING A
        WHERE A.DATETO < add_months (SYSDATE,-24);

        COMMIT;

    DELETE FROM A2_BOOKING WHERE DATETO < add_months (SYSDATE,-24);

        COMMIT;
END;

EXECUTE ARCHIVE_OLD_BOOKINGS;
```

	BOOKINGID	GUESTNO	DATEFROM	DATETO	ROOMID	ARCHIVED_DATE
1	1	1	10-OCT-17	20-OCT-15	1	01-NOV-17
2	8	4	30-OCT-17	08-NOV-12	24	01-NOV-17
3	11	6	23-OCT-17	03-JAN-14	36	01-NOV-17
4	12	6	10-OCT-17	08-NOV-12	37	01-NOV-17
5	14	7	05-OCT-17	08-NOV-12	39	01-NOV-17
6	18	3	17-OCT-17	08-NOV-13	9	01-NOV-17
7	20	5	17-OCT-17	08-NOV-12	13	01-NOV-17
8	22	7	17-OCT-17	08-NOV-12	35	01-NOV-17

	BOOKINGID	GUESTNO	DATEFROM	DATETO	ROOMID
1	2	1	29-OCT-17	15-NOV-17	14
2	3	2	03-OCT-17	29-NOV-17	15
3	4	2	29-AUG-17	17-OCT-17	16
4	5	3	05-OCT-16	28-OCT-16	17
5	6	3	24-OCT-17	16-NOV-17	28
6	7	4	26-OCT-17	30-NOV-17	25
7	9	5	29-OCT-17	15-NOV-17	26
8	10	5	28-OCT-17	28-DEC-17	27
9	13	7	01-OCT-17	02-DEC-17	38
10	15	1	17-OCT-17	08-NOV-17	2
11	16	2	17-OCT-17	08-NOV-17	3
12	17	2	17-OCT-17	08-NOV-17	7
13	19	4	17-OCT-17	08-NOV-17	12
14	21	6	17-OCT-17	08-NOV-17	33

Hand-ins: