60-415 ASSIGNMENT # 2 Solution (SQL DDL and PL/SQL)

Total: 8+5+(3+5+5)+6+6+6+6=50

1.

The EMP Table Structure Summary

EMP_NUM	CHAR(3) (Must be a
	number between 1 and
	1000) (Primary Key)
EMP_LNAME	VARCHAR2(15)
EMP_FNAME	VARCHAR2(15)
EMP_INITIAL	CHAR(1) (Must be a
	char between 'A' and
	'Z'
EMP_HIREDATE	DATE (NOT NULL)
JOB_CODE	VARCHAR2(10)
	(Foreign key to Job)

The JOB Table Structure Summary

JOB_ID	VARCHAR2(10) (Primary key)
JOB_TITLE	VARCHAR2(15) (NOT NULL)
MIN_SALARY	NUMBER(6)
MAX_SALARY	NUMBER(6)

Given this information, write a script called a2q1.sql to answer the following questions:

a. Write the SQL code that will create the table structures for Emp and Job .

```
CREATE TABLE emp(
           emp_num
                     CHAR(3) PRIMARY KEY,
           emp_lname VARCHAR2(15),
           emp_fname
                               VARCHAR2(15),
           emp_initial
emp_hiredate
                              CHAR(1),
                              DATE NOT NULL,
                               VARCHAR2 (10),
           job code
           constraint check_emp_num
                 CHECK (emp_num between 1 and 1000),
           constraint check emp initial
                 CHECK (emp_initial BETWEEN 'A' and 'Z')
           );
CREATE TABLE job(
            job_id
                              varchar2(10) PRIMARY KEY,
            job_title varchar2(15) NOT NULL,
            min_salary NUMBER(6),
            max salary NUMBER(6)
);
```

b. Write the SQL statement to add a column STARS(VARCHAR2(5) to the table JOB that has a default value of 1 * .

alter table job add stars varchar2(5) default '*';

c. Write a SELECT statement to display the constraints you created.

```
select constraint_type, constraint_type
from user_constraints
where table_name = &my_table_name;
```

(You can substitute my_table_name for JOB and EMP . Remember to give the table names in CAPITAL letters).

d. solution left to the reader !!

- 2. Build a PL/SQL block that computes the total compensation for one year.
 - a. The annual salary and the annual bonus percentage values are defined using the DEFINE command.
 - b. Pass the values defined in the above step to the PL/SQL block through SQL*Plus substitution variables. The bonus must be converted from a whole number to a decimal (For example from 15 to .15). If the salary is null, set it to zero before computing the total compensation. Execute the PL/SQL block. Reminder: Use the NVL function to handle NULL values.

3. a. Create a PL/SQL block that selects the maximum department number in the DEPARTMENTS table and stores it in an SQL*Plus variable.

```
PROMPT -----Assignment 2 Question 3a ------
VARIABLE g_max_deptno NUMBER

DECLARE

v_max_deptno NUMBER;

BEGIN

SELECT max(department_id)
INTO v_max_deptno
FROM departments;

:g_max_deptno := v_max_deptno;
```

END:

PRINT g_max_deptno

- b. Modife the PL/SQL block created in 4.a. that inserts a new department into the DEPARTMENT table.
 - i. use the DEFINE command to provide the department name. Name the new department Education.
 - ii. Pass the value defined for the department name to the PL/SQL block through a SQL*Plus substitution variable. Rather than printing the department number retrieved from 4.a., add 10 to it and use it as the department number and the new department.
 - iii. Leave the location number as null for now.
 - iv. Execute the PL/SQL block
 - v. Display the new department that u created (at the SQL prompt) .

```
PROMPT ----- Assignment 2 Question 3b -----
DEFINE p_dname = Education
DECLARE
    v_max_deptno departments.department_id%TYPE;
BEGIN
    SELECT MAX(department_id) + 10
    INTO v_max_deptno
    FROM departments;
    INSERT INTO departments
        (department_id, department_name, location_id)
        VALUES (v_max_deptno, '&p_dname', NULL);
    COMMIT;
END;
SELECT *
FROM departments
WHERE department_name='&p_dname';
```

- b. Create a PL/SQL block that deletes the department that you created in 3.a and b.
 - i. Use the DEFINE command to provide the department ID(DEFINE p_deptno=280)
 - ii. Pass the value to the PL/SQL through a SQL*Plus substitution variable.
 - iii. Print to the screen the number of rows affected.

```
PROMPT --Assignment 2 Question 4c ---

DEFINE p_deptno = 280

DECLARE

v_result NUMBER(2);

BEGIN

DELETE
FROM departments
WHERE department_id = &p_deptno;

v_result := SQL%ROWCOUNT;

DBMS_OUTPUT_PUT_LINE(TO_CHAR(v_result) ||' row(s) deleted ');

COMMIT;

END;
//
```

4. Use 3 substitution variables to store an employee number, the new department number and the percentage increase in the salary and Create a temporary table called emp which is a replica of table EMPLOYEES using the following commands:

```
DROP TABLE emp;
CREATE TABLE emp AS SELECT * FROM EMPLOYEES;
DEFINE P_EMPNO = 100
DEFINE P_NEW_DEPTNO = 20
DEFINE P_PER_INCREASE = 2
```

Update the department ID of the employee with the new department number, and update the salary with the new salary. Use the EMP table for the updates. After the update is complete, display the message, "Update complete" in the window. If no matching records are found, display "No Data Found."

```
DROP TABLE emp;
CREATE TABLE emp AS SELECT * FROM EMPLOYEES;
DEFINE p empno = 100
DEFINE p_new_deptno = 20
DEFINE p per increase = 2
PROMPT ---- Assignment 2 Question 4-----
BEGIN
    UPDATE emp
    SET department_id = &p_new_deptno,
      salary = salary + salary * 0.01 * &p_per_increase
    WHERE employee id = &p empno;
    IF SQL%FOUND THEN
        DBMS_OUTPUT.PUT_LINE('UPDATE COMPLETE');
    ELSE
        DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
    END IF:
    COMMIT;
END;
```

5. Create a PL/SQL block to declare a cursor EMP_CUR to select the employee name, salary, and hire date from the employees table. Process each row from the cursor, and if the salary is greater than 15,000 and the hire date is greater than 01-FEB-1988, display the employee name, salary, and hire date in the window in the format shown in the sample output below:

Kochhar earns 17000 and joined the organization on 21-SEP-89 De Haan earns 17000 and joined the organization on 13-JAN-93 PL/SQL procedure successfully completed.

```
PROMPT ---- Assignment 2 Question 5 ----
SET SERVEROUTPUT ON
DECLARE
   CURSOR EMP CUR IS
        SELECT last name, salary, hire date FROM EMPLOYEES;
    ENAME VARCHAR2(25);
    SAL NUMBER(7,2);
    HIREDATE DATE;
BEGIN
   OPEN EMP_CUR;
    FETCH EMP_CUR INTO ENAME, SAL, HIREDATE;
    WHILE EMP CUR%FOUND
    LOOP
        IF SAL > 15000 AND HIREDATE >= TO DATE('01-FEB-1988','DD-
MON-YYYY') THEN
            DBMS OUTPUT.PUT LINE (ENAME | 'earns' |
                TO_CHAR(SAL)|| ' and joined the organization on ' ||
                TO_DATE(HIREDATE, 'DD_MON_YYYY'));
        END IF:
    FETCH EMP CUR INTO ENAME, SAL, HIREDATE;
    END LOOP:
    CLOSE EMP CUR;
END;
```

6. a. Create a PL/SQL block that declares a cursor called DATE_CUR. Pass a parameter of DATE data type to the cursor and print the details of all the employees who have joined after that date.

DEFINE P_HIREDATE = 08-MAR-00

b. Test the PL/SQL block for the following hire dates:

08-MAR-00, 25-JUN-97, 28-SEP-98, 07-FEB-99.

PROMPT ---- Assignment 2 Question 6----

```
DECLARE
    CURSOR DATE_CURSOR(JOIN_DATE DATE) IS
        SELECT employee_id,last_name,hire_date FROM employees
        WHERE HIRE DATE > JOIN DATE;
    EMPNO employees.employee id%TYPE;
    ENAME employees.last_name%TYPE;
    HIREDATE employees.hire date%TYPE;
    HDATE employees.hire_date%TYPE := '&P_HIREDATE';
BEGIN
    OPEN DATE CURSOR(HDATE);
    LOOP
     FETCH DATE CURSOR INTO EMPNO, ENAME, HIREDATE;
     EXIT WHEN DATE_CURSOR%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE (EMPNO || ' ' || ENAME || ' ' || HIREDATE);
   END LOOP;
END;
```

7. In a loop, use a cursor to retrieve the department number and the department name from the DEPARTMENTS table for those departments whose DEPARTMENT_ID is less than 100. Pass the department number to another cursor to retrieve from the EMPLOYEES table the details of emplyee last name, job, hire date, and salary of those employees whose EMPLOYEE_ID is less than 120 and who work in that department.

```
PROMPT ---- Assignment 2 Question 7----
DECLARE
    CURSOR dept_cursor IS
        SELECT department_id, department_name
        FROM departments
        WHERE department_id < 100
        ORDER BY department id;
    CURSOR emp_cursor(v_deptno NUMBER) IS
        SELECT last_name, job_id, hire_date, salary
        FROM employees
        WHERE department id = v deptno
        AND employee id < 120;
    v current deptno
                       departments.department id%TYPE;
                        departments.department_name%TYPE;
    v_current_dname
    v ename
                     employees.last name%TYPE:
                   employees.job id%TYPE;
    v job
                    employees.hire date%TYPE;
    v hiredate
                  employees.salary%TYPE;
    v sal
    v_line
                  varchar2(100);
BEGIN
    OPEN dept cursor;
    LOOP
        FETCH dept_cursor INTO v_current_deptno, v_current_dname;
        EXIT WHEN dept_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Department Number : '||
                                   v_current_deptno ||
                                   Department Name: ' ||
                                  v_current_dname);
        IF emp_cursor%ISOPEN THEN
            CLOSE emp_cursor;
```

```
END IF;
        OPEN emp_cursor(v_current_deptno);
        LOOP
            FETCH emp_cursor INTO v_ename, v_job, v_hiredate, v_sal;
            EXIT WHEN emp_cursor%NOTFOUND;
             DBMS_OUTPUT.PUT_LINE( v_ename ||
                                        v_job ||
                                        v_hiredate||
                                        ' '||
                                        v_sal);
        END LOOP;
        IF emp_cursor%ISOPEN THEN
            CLOSE emp_cursor;
        END IF;
    END LOOP;
    IF emp_cursor%ISOPEN THEN
            CLOSE emp_cursor;
    END IF;
    CLOSE dept_cursor;
END;
```