**Homework 3 Solution**

**Maximum:** 100pts  
**Note:** This homework is to be done by each student individually.   
**Purpose of homework:** To become familiar with the foundations of relational database querying

**Problem 1: Getting Started with Oracle**[30 pts]

In SQLDeveloper, create table:

CREATE TABLE SuperBallScores (  
playerNum INTEGER,  
score INTEGER  
);

Set aside the SQLDeveloper window and open your Python IDE and copy the following code :-

*import random*

*import csv*

*with open("sample.csv", "w", newline="") as csvfile:*

*csvwriter = csv.writer(csvfile)*

*csvwriter.writerow(["PLAYERNUM", "SCORE"])*

*for i in range(50000):*

*roll = random.randint(1, 1000)*

*csvwriter.writerow([i + 1, roll])*

The execution of this script will create sample data in a file named "sample.csv".

Return to your SQLDeveloper window and find the table that you created in the left-hand side column and right-click the table. Then click on Import Data.

After successful importing data, you are ready to run your SQL statements.

Also, for SQLDeveloper, if we execute the command "SET TIMING ON;", you will get the time it took the database server to execute each given SQL statement. For each of the following SQL statements, run them, and then report the answer, as well as the time it took for executing that statement. Lastly please observe which query ran the quickest, which the slowest, and so on. Then indicate if this is what you had expected that would happen, or if you are surprised. Explain your thoughts about your observations.

* SELECT COUNT(playerNum) FROM SuperBallScores;
* SELECT playerNum, MIN(score), MAX (score) FROM SuperBallScores GROUP BY playerNum;
* SELECT T1playerNum FROM SuperBallScores T1, SuperBallScores T2 WHERE T1.playerNum = T2.playerNum;
* UPDATE SuperBallScores SET score=score - 10;

The data is huge. Thus, do NOT submit the actual log showing the loading of the data or the execution of the queries. Instead, for each of the above queries simply report the performance results (numbers) you observed.

Below are examples of what to submit.

SQL> SELECT COUNT(playerNum) FROM SuperBallScores;

COUNT(PLAYERNUM)

----------------

49569

Elapsed: 00:00:00.03

SQL> SELECT playerNum,MIN(score),MAX(score) FROM SuperBallScores Group By playerNum;

49569 rows selected.

Elapsed: 00:00:01.71

SQL> SELECT T1.playerNum FROM SuperBallScores T1,SuperBallScores T2 WHERE T1.playerNum=T2.playerNum;

49569 rows selected.

Elapsed: 00:00:01.80

SQL> UPDATE SuperBallScores SET score=score-10;

49569 rows updated.

Elapsed: 00:00:01.09

**Problem 2: Relational Algebra About Employment.** [70 pts]

Relation Emp

eid Integer,

ename Char Width 255,

age Integer,

salary Float

PRIMARY KEY (eid)

relation Works

eid Integer,

did Integer,

pct\_time Integer

PRIMARY KEY (eid,did),

FOREIGN KEY (eid) REFERENCES Emp (eid),

FOREIGN KEY (did) REFERENCES Dept (did);

relation Dept

did Integer,

dname Char Width 255,

budget Float,

managerid Integer,

PRIMARY KEY (did)

For simplicity and readability, below I have broken the expressions into smaller subexpressions and assigned intermediate relation names to each. We can, however, form one expression by substituting the definition of each expression in for its actual expression name.

1.

RENAME(tmp1, PROJECT[eid] (SELECT[dname='hardware'](Dept NAT-JOIN Works )))

RENAME(tmp2, PROJECT[eid] (SELECT[dname='software'](Dept NAT-JOIN Works )))

PROJECT[ename,age] ( (tmp1 INTERSECT tmp2 ) NAT-JOIN[on eid] Emp)

2.

PROJECT[did, count-emp] (SELECT[parttime-sum>1000]

(GROUP [did] [(sum(pct-time) as parttime-sum); (count(\*) as count-emp)] (Dept)))

3.

PROJECT[ename](Emp) – set difference

PROJECT[ename](SELECT[sal<=budget](Emp NAT-JOIN[eid]

Works NAT-JOIN[did] Dept))

4.

PROJECT DISTINCT[managerid](Dept) – set difference

PROJECT[managerid](SELECT[budget<=1,000,000]] (Dept ))

5. We utilize the aggregate operator below.

RENAME(tmp1, GROUP [ ] [(budget: max)] (Dept ))

SELECT[ename] (Emp NAT-JOIN[on eid] (Dept NAT-JOIN[budget=max-budget] tmp1 ))