Tutorial 1
SQL Exercises

Solutions

Notes:

a. This tutorial consists of 2 parts (Part A: Managing Tables, and Part B: Querying Exercises). You are required to attempt all parts. You should spend on average 4 minutes on each question.
b. To answer the questions in this tutorial, you can solely refer to the lecture notes posted in Moodle (Week-1d-SQL-Revision.ppt).
c. All the questions needs to be completed using SQL Query. You could cut and paste the SQL codes, if they are given. However, when you cut and paste the codes from Microsoft Word, be careful with the quote ('), as sometimes it may not be compatible with the text format in SQL.
d. If you have not finished all the questions, you should complete them before the start of tute-2. Don't get behind!!

PART A. Managing Tables

1. After a successful login to your Oracle account using SQL Developer, type in the following SQL statement in the SQL Developer window:

```
SELECT * FROM TAB;
```

   Write down your observation(s).

2. Type in the following SQL statement

```
CREATE TABLE LECTURER
(StaffNO                NUMBER(6)        NOT NULL,
 Title                  VARCHAR2(3),
 FName                  VARCHAR2(30),
 LName                  VARCHAR2(30),
 StreetAddress          VARCHAR2(70),
 Suburb                 VARCHAR2(40),
 City                   VARCHAR2(40),
 PostCode               VARCHAR2(4),
 Country                VARCHAR2(30),
 LecturerLevel          CHAR(2),
 BankNO                 CHAR(20),
 BankName               VARCHAR2(40),
 Salary                 NUMBER(8,2),
 WorkLoad               NUMBER(2,1)      NOT NULL,
 ResearchArea           VARCHAR2(40),
 PRIMARY KEY(StaffNo));
```

3. Type in the following SQL statement:

```
SELECT * FROM TAB;
```

   Write down your observation(s).

4. Type in the following SQL statements

a)

```
INSERT INTO LECTURER (StaffNO, Title, FName, LName,
   StreetAddress, Suburb,City, PostCode, Country, LecturerLevel,
   BankNO, BankName, Salary, WorkLoad, ResearchArea)
VALUES (1000,'Dr','David','Taniar','3 Robinson Av', 'Kew',
   'Melbourne', '3080', 'Australia', '5', '1000567237',
   'CommBank', 89000.00, 2.0, 'O-R DB');
```

b)

```
INSERT INTO LECTURER (StaffNO, Title, FName, LName,
   StreetAddress, Suburb, City, PostCode, Country,
   LecturerLevel, BankNO, BankName, Salary, WorkLoad,
   ResearchArea)
VALUES (1000,'Ms','Julie','Main','6 Algorithm Av',
   'Montmorency', 'Melbourne', '3089', 'Australia', '5',
   '1000123456', 'CommBank', 89000.00, 2.0, 'CBR');
```

What happens? Why?
Correct the mistake of the last SQL command by changing the staffNO to 2000.

c) Note: if you are entering values for all attributes, as in a) and b) above, you do not need to include the attribute names. Enter the following SQL statement:

```
INSERT INTO LECTURER VALUES (3000, 'Mr', 'Daniel', 'Wright',
   '22 Crystal Cres', 'Alphington', 'Melbourne', '3790',
   'Australia', '5', '1000654321', 'CommBank', 89000.00, 2.0,
   'DB');
```

d) However, if you are only entering partial information you MUST specify the attributes and ensure that all NOT NULL fields have a value. Enter the following SQL statement:

```
INSERT INTO LECTURER (StaffNO, Title, FName, LName,
   StreetAddress, Suburb, PostCode, Country, ResearchArea,
   Workload)
VALUES (4000, 'Mr', 'RaiHong', 'Lam', '12 Oracle Dr',
   'Fitzroy', '3424', 'Australia', 'Data Mining', 1);
```

5. Type in the following SQL statement:

```
SELECT * FROM LECTURER;
```

Write down your observation(s).

6. a) Type in the following SQL statement. Note that in the following SQL statement, the city is spelt with the case as follows: **CiTTy**

```
CREATE TABLE STUDENT
(StudentNO            NUMBER(6)  NOT NULL,
 DOB                  DATE,
 FName                VARCHAR2(30),
 LName                VARCHAR2(30),
 -- city spelt CiTTy
 CiTTy                VARCHAR2(40),
 PostCode             VARCHAR2(4),
 Country              VARCHAR2(30),
 FeePaid              NUMBER(8,2),
 LastFeeDate          DATE,
 PRIMARY KEY(StudentNo));
```

   b) Insert 5 student into the student table, with student number 30001, 30002, 30003, 30004 and 30005. Assign all attributes values. Note format for inserting date: '12-FEB-2002'.

Notes:
To Insert Records:

Format for inserting date and time into a table:

```
TO_DATE('12-MAR-2001 16:15', 'DD-MON-YYYY HH24:MI')
```

To Display/Retrieve Records:

Format for displaying date and time:

```
TO_CHAR(NameOfAttribute, 'DD-MON-YYYY HH24:MI')
```

For example:

```
SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI')
FROM DUAL;
```

7. Type in the following SQL statement:

```
ALTER TABLE STUDENT ADD
(StreetAddress          VARCHAR2(70),
 Suburb                 VARCHAR2(40));
```

8. Type in the following SQL statement:

```
DESCRIBE STUDENT;
```

Or

```
DESC STUDENT;
```

Write down your observation(s).

9. Type in the following SQL statement:

```
ALTER TABLE STUDENT
DROP(CiTTy);
```

10. Type in the following SQL statement:

```
ALTER TABLE STUDENT
ADD (City     CHAR(40));
```

11. Type in the following SQL statement:

```
ALTER TABLE STUDENT
MODIFY (City  VARCHAR2(40));
```

Explain what the difference between CHAR and VARCHAR2.

CHAR is a fixed length string data type. In Question#10 above, if the length of a city is less than 40 characters, the system will pad with spaces in order to fill up the 40 character spaces allocated to the City attribute.

VARCHAR2 is a variable length string data type. In Question#11, if the length of a city is less than 40 characters, the system will not pad with additional spaces at the end. Instead, the left over spaces will be immediately used by the next attribute.

12. Type in the following SQL statement:

```
UPDATE STUDENT
SET StreetAddress = '12 New St'
WHERE StudentNo = 30001;
```

Display the contents of the Student table and write down your observation(s).

13. Can you ADD a new field and DROP another field in one SQL Statement?

Explain your answer.

No. One Alter Table statement for Adding new attributes, and a separate Alter Table statement for deleting attributes.

14. Type in the following SQL statement and explain what happens.

```
COMMIT;
```

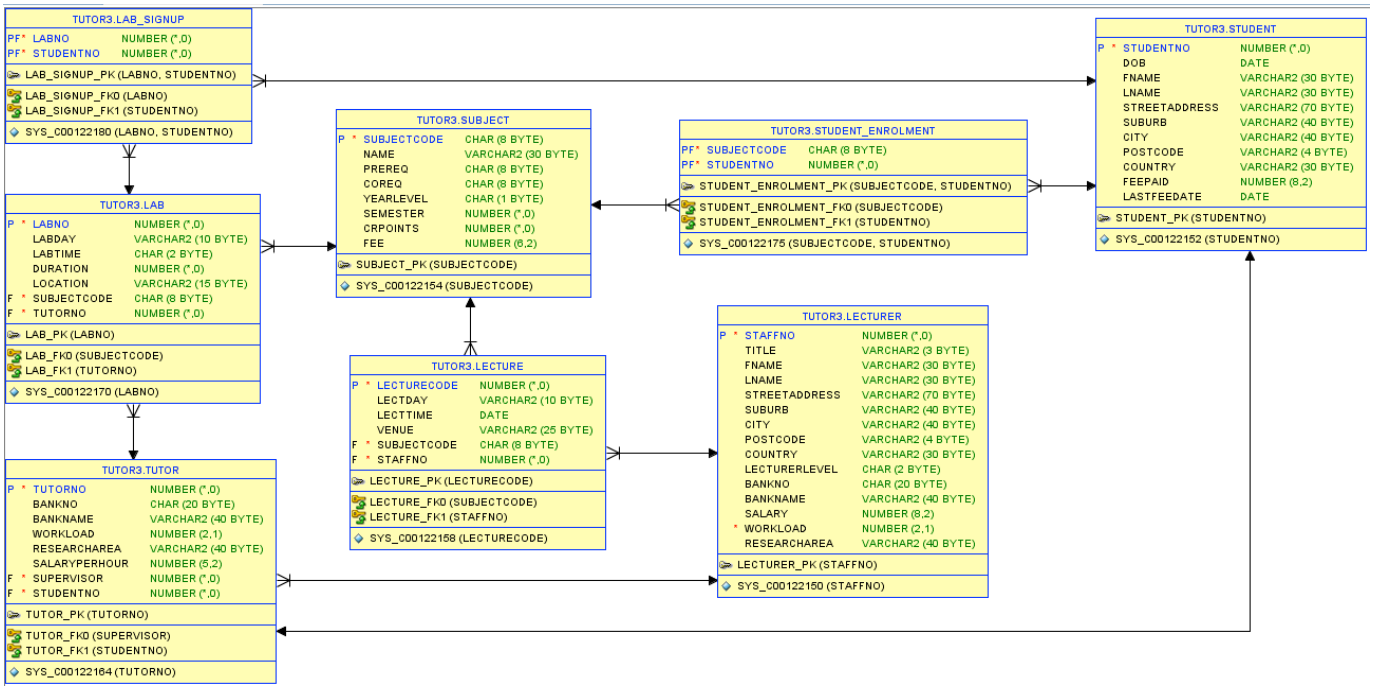
PART B. Querying Exercises

15. You should have 2 tables (Lecturer and Student tables) in your account where you have created in Part A (Managing table).

16. Tables SUBJECT, LECTURE, TUTOR, LAB, STUDENT_ENROLMENT, and LAB_SIGNUP have been created in the `dtaniar` account. Several records have been inserted to this table. You can now import the tables into your account using the following SQL statement, for e.g.:

```
Create Table SUBJECT
As Select *
From dtaniar.SUBJECT;
```

You need to import all other tables (i.e. LECTURE, TUTOR, LAB, STUDENT_ENROLMENT, and LAB_SIGNUP). For your reference, the E/R diagram of these tables is shown below:




```
Create Table LECTURE
As Select *
From dtaniar. LECTURE;

Create Table TUTOR
As Select *
From dtaniar. TUTOR;

Create Table LAB
As Select *
From dtaniar. LAB;
```

```
Create Table STUDENT_ENROLMENT
As Select *
From dtaniar. STUDENT_ENROLMENT;

Create Table LAB_SIGNUP
As Select *
From dtaniar. LAB_SIGNUP;
```

17. Write the SQL statement to list all the lecturers and their lecture schedules

```
SELECT      L.LName, L.FName, S.SubjectCode, S.LectDay, S.LectTime
FROM        Lecturer L, Lecture S
WHERE       L.StaffNo = S.StaffNo;
```

18. Are there any lecturers who not teaching?

```
SELECT      FName, LName
FROM        Lecturer
WHERE       StaffNo NOT IN (
                    SELECT      StaffNo
                    FROM        Lecture
);
```

19. List all the subjects offered in the first semester.

```
SELECT      *
FROM        Subject
WHERE       Semester = 1;
```

20. List all the students by first-name, last-name, date-of-birth, and fee-paid details, who are born after 1990 and before 1995.

```
SELECT      FName, LName, DOB, FeePaid
FROM        Student
WHERE       DOB BETWEEN to_date('01-JAN-1991','DD-MON-YYYY')
            AND to_date('31-DEC-1994','DD-MON-YYYY');
```

Alternatively:

```
SELECT      FName, LName, DOB, FeePaid
FROM        Student
WHERE       to_char(DOB, 'YYYYMMYY') >= '19910101'
            AND to_char(DOB, 'YYYYMMYY') <= '19941231';
```

Or, you could just retrieve the year:

```
SELECT      FName, LName, DOB, FeePaid
FROM        Student
WHERE       DOB BETWEEN TO_DATE('1991','YYYY')
```

```
AND             TO_DATE('1994','YYYY');
```

Or

```
SELECT      FName, LName, DOB, FeePaid
FROM        Student
WHERE       to_char(DOB, 'YYYY') > '1990'
            AND to_char(DOB, 'YYYY') < '1995';
```

21. List all the students enrolled in the database subject. (Note: database = CSE21DB, CSE31DB, CSE41FDB)

```
SELECT      DISTINCT S.StudentNo, FName, LName
FROM        Student S, Student_Enrolment SE
WHERE       (SubjectCode = 'CSE21DB'
OR          SubjectCode = 'CSE31DB'
OR          SubjectCode = 'CSE41FDB')
AND         S.StudentNo = SE.StudentNo;
```

Alternatively:

```
SELECT      DISTINCT S.STUDENTNO, S.FNAME, S.LNAME
FROM        Student S, Student_Enrolment SE
WHERE       SE.SUBJECTCODE IN ('CSE21DB', 'CSE31DB', 'CSE41FDB')
AND         S.STUDENTNO=SE.STUDENTNO;
```

Or:

```
SELECT      DISTINCT S.StudentNo, FName, LName
FROM        Student S, Student_Enrolment SE
WHERE       SubjectCode LIKE '%DB%'
AND         S.StudentNo = SE.StudentNo;
```

22. List the students who are tutors.

```
SELECT      T.TutorNo, T.StudentNo, FName, LName
FROM        Student S, Tutor T
WHERE       S.StudentNo = T.StudentNo;
```

23. Select the lecturer(s) whose research area is ‘Network Management’.

```
SELECT      StaffNo, FName, LName
FROM        Lecturer
WHERE       ResearchArea = 'Network Management';
```

24. Calculate the average salary of a lecturer.

```
SELECT      AVG(Salary) AS “Average Salary”
FROM        Lecturer;
```

Notes: NULL values will not be included in the average calculation. If you would include the NULL values into the average calculation, then you could use the NVL function. In this example, if Salary is NULL then Salary=0.

```
SELECT      AVG(NVL(SALARY,0)) AS "Average Salary"
FROM        LECTURER;
```

25. Calculate the minimum and maximum salary of the lecturers.

```
SELECT      MIN(Salary) AS “Min Salary”, MAX(Salary) AS “Max Salary”
FROM        Lecturer;
```

26. List the number of tutors by each subject and semester.

```
SELECT      SJ.Name as SubjectName, SJ.Semester, COUNT(TT.TutorNo)
FROM        Subject SJ, Lab LB, Tutor TT
WHERE       SJ.SubjectCode = LB.SubjectCode
AND         TT.TutorNo = LB.TutorNo
GROUP BY    SJ.Name, SJ.Semester;
```

Alternatively:

```
SELECT      SJ.Name as SubjectName, SJ.Semester, COUNT(LB.TutorNo)
FROM        Subject SJ, Lab LB
WHERE       SJ.SubjectCode = LB.SubjectCode
GROUP BY    SJ.Name, SJ.Semester;
```

27. List the total number of students in each lab, for each subject, with the tutor’s name.

```
SELECT      SJ.Name as SubjectName, LB.LabNo, ST.LName as Tutor,
            COUNT(LS.StudentNo) as NumberOfStudent
FROM        Subject SJ, Lab LB, Lab_SignUp LS, Tutor TT, Student ST
WHERE       SJ.SubjectCode = LB.SubjectCode
AND         LB.LabNo = LS.LabNo
AND         TT.TutorNo = LB.TutorNo
AND         TT.StudentNo = ST.StudentNo
GROUP BY    SJ.Name, LB.LabNo, ST.LName;
```

28. Calculate the cost of running all the database labs per week. (Hint: lab duration * tutors' SALARYPERHOUR)

```
SELECT    SUM(TT.SalaryPerHour * LB.Duration) as "Database Labs Cost Per Week"
FROM      Lab LB, Tutor TT, Subject SJ
WHERE     LB.SubjectCode = SJ.SubjectCode
AND       LB.TutorNo = TT.TutorNo
AND       SJ.SubjectCode LIKE '%DB%';
```

Alternatively:

```
SELECT    SUM(TT.SalaryPerHour * LB.Duration) as "Database Labs Cost Per Week"
FROM      Lab LB, Tutor TT
WHERE     LB.TutorNo = TT.TutorNo
AND       LB.SubjectCode LIKE '%DB%';
```

The End