SCHOOL OF
COMPUTER AND SECURITY SCIENCE

AUSTRALIA ECU EDITH COWAN UNIVERSITY

CSG1207/CSI5135: Systems and Database Design
Lab 08

Introduction

This lab allows you to practise the SQL covered in this week's lecture. Do your best to answer the following questions and write the specified queries. You are encouraged to experiment with SQL – it is a very flexible language, so if you can think of something that would be useful to achieve in a query, it can probably be done. This lab uses the "company" database, which you can create by running the script file Module 5 of the unit materials. It is assumed that you are working in SQL Server Management Studio, also covered in Module 5.

If you are having trouble writing an SQL query, read any error messages and try to fix the error. Search for examples in the unit materials, and ask your tutor for assistance if needed. Contact your tutor if you spot something which appears to be incorrect in any of the labs.

Lab Tasks

Q1. Insert a new row into the "job" table, with the following details. For this insert, do not list the column names after the name of the table.

job_id	job_title	min_salary	max_salary
GN_SEC	Secretary	3500	10000

Q2. Insert another row into the "job" table, with the following details. This time, list the column names after the name of the table.

job_id	job_title	min_salary	max_salary
GN_JAN	Janitor	1500	NULL

Q3. Insert another row into the "job" table, with the following details. You can choose whether or not to list the column names this time.

job_id	job_title	min_salary	max_salary
GN_CAF	Cafeteria Worker	DEFAULT	4500

Q4. Write an UPDATE statement which updates rows in the "job" table, giving the GN_SEC and GN_JAN jobs a maximum salary of 5000.

Q5. *This task uses a multiple-insert statement, which is not supported in SQL Server 2005. If you are using SQL Server 2005, skip this task.*
The following statement is trying to insert three rows into the "country" table using a multiple-insert, however "33" was accidentally typed instead of "3". Run the statement without fixing it – does SQL Server insert the correct rows, or none of them?

```
INSERT INTO country
VALUES ('CH', 'China', 3),
       ('JP', 'Japan', 33),
       ('EG', 'Egypt', 4);
```

Then, fix the typo and run the statement to insert the three countries.

Q6. Write a DELETE statement that deletes any countries from the "country" table that have a region ID larger than 2.

Q7. Write an UPDATE statement that uses a subquery to update the maximum salary of the GN_CAF job to be the same as the maximum salary of the GN_SEC job.

Q8. Write statements to do the following (do not run them until you have written them all):

1. *Begin a transaction* (it does not need a name).

2. *Insert a new row in the "job" table* with the following values:

job_id	job_title	min_salary	max_salary
GN_SPY	Corporate Spy	50000	75000

3. *Create a save point in the transaction* named "after_spy".

4. *Update all rows in the "job" table* to have a minimum salary of 50000 and a maximum salary of 75000.

5. *Select all data from the "job" table* to see the current state of the data.

6. *Roll back the transaction* to the "after_spy" save point.

7. *Commit the transaction*.

Once you have written and run the complete transaction, select all data from the "job" table to see which changes were committed. Which changes have been committed?

Q9. Write a DELETE statement that deletes all jobs with a job ID starting in "GN".

Challenge Query!

The following query is more difficult than the previous queries, and may involve SQL syntax which has not yet been covered in the unit.

Q10. This week's challenge query practices inserting data from one table into another using a subquery, covered on slide 10 of Module 8's lecture. First, create a table with the following specifications:

"emp_summary" table				
Column Name	**Data Type & Length**	**Null**	**Constraints**	**Other**
emp_id	INT	NOT NULL	PK	
full_name	VARCHAR(50)	NOT NULL		
email	VARCHAR(75)	NULL		
phone	VARCHAR(20)	NULL		

Once you have create the table, we need to populate it with the appropriate data from the "employee" table. The corresponding columns are as follows:

employee table		**emp_summary** table
`employee_id`	▶	`emp_id`
`first_name + ' ' + last_name`	▶	`full_name`
`LOWER(email) + '@company.com'`	▶	`email`
`phone_number`	▶	`phone`

Write an INSERT statement that uses a subquery to insert the data from the "employee" table into the "emp_summary" table following the specifications above. Check that you have copied the data correctly by selecting all rows from "emp_summary".

Note: What we have done is a lab exercise only. It has duplicated data, which goes against the purpose and goals of a database. It would be much more effective to create a view, covered in Module 10.