School of Computer Science and IT, RMIT

ISYS1055/1057 Tutorial/Lab Sheet

SQL Programming (1)

A read-only version of the Rocky Concrete database has been built up for you to complete exercises. The database schema and content are described in Attachment 1. Attachment 2 contains description on some basic Oracle SQL*Plus commands, and how to create a .sql file containing all queries.

Complete the following questions using the read-only Rocky Concrete database, which is accessed by the **sqlrocky** command on Yallara. Write an SQL query for each question below. Make sure that each query is ended with a ";" and "/". Your queries must compile and run under "sqlrocky".

Put all your SQL queries in a .sql file named after your student number. For example, if your student number is S1234, your file containing SQL queries should be called S1234.sql. Compress this file and the .pdf file for your last tute into a .zip file and submit .zip file to Practice Assignment 1 in WebLearn. For example, assuming that S1234.pdf and S1234.sql are two files, the following command compress these files into S1234.zip:

```
zip S1234.zip S1234.pdf S1234.sql.
```

1. Display the name, adddress (street, town, postcode), and credit limit of customer "Di Hunter".
2. Calculate the total number of customers in each town. Output should be a list of town and total-number-of-customers pairs, in alphabetical order of town.
3. The account officer of Rocky Concrete needs to regularly find customers who reach their credit limit and issue a notice of payment. A customer reaches his/her credit limit if the difference between his/her credit limit and current balance is no more than $50. Which customers have reached their respective credit limit? List the name and address of these customers.
4. For each group of products, calculate the average price for the group. Output the product groups and their average product price.
5. Produce a list of product groups where each group has a total product value of at least $2000. The total product value should also be output for each of these groups.
6. List the product code ("prod_cod") of the "Tank" and "Trough" products (description contains the word "Tank" or "Trough").

Attachment 1.

The Rocky Concrete database Schema and Instance

Customers(<u>cust no</u>, cust_name, street, town, postcode, cr_limit, curr_balance)

Products(<u>prod cod</u>, description, prod_group, list_price, qty_on_hand,
remake_level, remake_qty)

Orders(<u>order no</u>, order_date, cust_no*)

Order_details(<u>order_no*, prod cod*</u>, order_qty, order_price)

cust_no	cust_name	street	town	postcode	cr_limit	curr_balance
1066	Nevs Nursery	White Hart	Bundoora	3083	500	450
13144	Preston City	High Street	Preston	3072	3000	1000
1776	Di Hunter	Thornton Farm	Whittlesea	3757	500	500
2001	Glads Gladdies	Childs Road	Mill Park	3082	500	0
2002	Mill Park	Betula Ave	Mill Park	3082	1000	300

prod_cod	description	prod_group	list_price	qty_on_hand	remake_level	remake_qty
MOO	Medium Cattle Trough	A	150	6	3	5
LOO	Large Cattle Trough	A	250	1	1	3
STANK	Small Septic Tank	D	300	10	5	15
LTANK	Large Septic Tank	D	450	1	2	2
LTUB	Laundry Tub	D	100	20	15	20
GNOME	Garden Gnome	D	10	100	150	200
STAND	Bicycle Stand	C	50	50	35	20
GABBY	Football Player Statue	C	500	10	15	40

order_no	order_date	cust_no
1	1/07/1993	13144
2	2/07/1993	13144
3	2/07/1993	1066

order_no	prod_cod	order_qty	order_price
1	STAND	10	45
1	GABBY	2	480
2	STAND	5	45
3	GNOME	10	10

Attachment 2.

On Using Oracle SQL*Plus

Once you have started the Oracle SQL*Plus engine (when you see the "SQL>" prompt), there are two types of constructs you can execute:

- SQL*Plus commands – get executed immediately.
- SQL programs (queries) – get executed when ending with ";" or "/".

What are the basic SQL*Plus commands?

Below is a list of some frequently used SQL*Plus commands (not in a particular order). When applicable, examples are given using the Rocky Concrete database.

- HELP – access the help system for SQL*Plus command (but not SQL programming!!). Examples:
    - `Help index` --- list all help items.
    - `help save` – display help info for the command "save".
- START - Run a SQL script file (short: @)
- EXIT or QUIT - Disconnect from the database and terminate SQL*Plus
- SAVE – save the content of the SQL buffer into a file. Options for the command: *create*, *replace* or *append*. For example:
    - `save s1234.sql append`
- DESCRIBE - Lists the attributes of tables and other objects (short: DESC)
    - `desc Customers` --- describe the Customers table
- EDIT - Places you in an editor so you can edit a SQL command (short: ED)
- LIST - Displays the last command executed/ command in the SQL buffer (short: L)
- SET - Modify the SQL*Plus environment. Setting some display variables are very useful for elegant output. For example,
    - `set pagesize 100` --- maximal 100 rows for a page.
    - `set linesize 100` --- maximal 100 characters for a line
- COLUMN – set the display format for attributes (columns)
    - `column cust_no format 9999` --- 4-digit wide integer
    - `column cust_name format a15` – 15-character wide string
    - `column street format a10` – 10 character wide string
    - `column postcode format 99` – 2-digit wide string

What is AFIEDT.BUF?

AFIEDT.BUF is the SQL*Plus default edit save file. When you issue the command "ed" or "edit" without arguments, the last SQL statements will be saved to a file called AFIEDT.BUF and opened in the default editor.

Compose a .sql file keeping SQL queries.

The default editor for the SQL*Plus SQL command buffer (the last SQL statement entered but not the SQL*Plus command!!) with the EDIT (or ED) command is "vi". You can set the editor another text editor by setting the DEFINE_EDITOR variable.

For example, the following example set the default editor to “pico”, which can be used in a non-graphical environment and very user-friendly.

```
DEFINE _EDITOR=pico
```

Composing an SQL query often involves (often several rounds of) editing, debugging, and trial execution. To do this the following steps (not in a particular order) are often used in this process:

- Set the default editor to “pico”:
  ```
  DEFINE_EDITOR=pico
  ```
- At the “SQL>” prompt, type in an SQL query. When complete execute the query. If there are errors, type in the “edit” command to invoke the “pico” editor to edit the query.
- In editing an SQL query using “pico”, when editing finishes, type Ctrl+X to exit the system, and save the buffer to afiedt.buf. Note that you should not give another file name.
- At the “SQL>” prompt, type in the “list” command to check if the query in system buffer is what you what.

When an SQL query is finalised after editing and debugging, at the “SQL>” prompt use the “save” command to append and save the query in a file. For example, the following command append and save the query in buffer in the file S1234.sql.

```
save S1234.sql append
```

After several rounds of composition, editing and saving operations, a .sql file is created that contains all SQL queries that have been in the system buffer. The .sql file can be further edited using “pico” from the Unix system prompt. Especially, comments can be added. Comments are delimited by a pair of “/*” and “*/”. Alternatively a line starting with “//” is a comment line.

Shown below is a sample .sql file that contains two SQL queries and comments.

```
// This is the first query.
Select *
From Customers
/

/*
 * This coment uses several lines.
 * This is just an example.
 * This is trying an aggregation query.
  */

select town, avg(cr_limit)
from Customers
group by town
/
```