ICT394 Business Intelligence Application Development

# Lab 04: SQL for BI

Version 1.0: 8th February 2016

* Updated 22nd February 2016
* Updated 18th February 2017

Version 2.0 28th August 2017

* Updated 2nd September 2017

Version 3.0 22nd January 2018

* Updated 30th April 2021
* **Updated 12th September 2022**

## For this lab, you will need:

* An ssh client (e.g., PuTTy) –
* SQL Developer (optional) – SQL Developer is installed in the Computer Labs
* Cisco AnyConnect VPN. You MUST connect to VPN first BEFORE logging into PuTTy and SQL Developer
* PuTTy, Cisco AnyConnect VPN and SQL Developer are installed in the Computer Labs

## Introduction

**In your previous database studies**, you have met some of the **analytic functions provided in SQL** (Oracle SQL in particular), including PIVOT/UNPIVOT, LAG/LEAD, and SUM/RANK. There are also other SQL operators that are used in data warehousing including ROLLUP, CUBE, and the various GROUPING functions.

An **understanding of how these functions work** is **invaluable in creating BI** queries and reports. **In this lab**, you will **use the Oracle database** on **rhea.ad.murdoch.edu.au**. You can use either a shell or SQL Developer.

## Logging into Oracle SQLplus

### LOGGING INTO ORACLE SQLplus using SSH

**Please, please, please – BEFORE you send me an email that you cannot log into Oracle, make sure you have followed the instructions below.**

The first time you log into Oracle SQLplus, you **MUST** do so using an SSH client such as Putty or using the SSH command in a terminal session in Linux or MacOS[[1]](#footnote-1). Oracle is located on the linux server **rhea.ad.murdoch.edu.au**, and **to use Oracle** you need to **log into rhea first** then to the Oracle database.

Putty can be downloaded from <http://www.chiark.greenend.org.uk/~sgtatham/putty/> . Please note that **ANY SSH client[[2]](#footnote-2)** can be used, though it is useful if the client used has some way of either logging the session or piping output to a file.

* **Start putty** and **enter rhea.ad.murdoch.edu.au into the hostname.** Make sure you **select SSH** as the **protocol.**

Graphical user interface, application

Description automatically generated

Several security warnings may appear – you are quite safe to answer ‘Yes’. The session window should now open and you will see the login prompt for **rhea**:

Text

Description automatically generated with medium confidence

* **Enter your student number and university password to log into rhea.**

You are now logged in to rhea and see the rhea prompt.

The Oracle component we are using is called SQLPlus.

* **To log in to Oracle SQLPlus, type sqlplus at the rhea prompt.**

SQLPlus starts and you will be prompted for **your Oracle username** and **password**:

Text

Description automatically generated

**Username and password →** Your **ORACLE SQLPLUS** username is an uppercase A followed by your student number. For example, if your student number was **20012001**, your username and initial password would be **A20012001**

**Type in your username and type in the same again for your password.**

The first time you log on **you will be prompted to change your password**. You will see a message stating that the account has expired and you need to **change your password**.

* **At the prompts enter your old password (the one you just entered), then your new password, then your new password again.**

Your password must be at least 6 characters and contain alphabetical and numerical characters, eg. h3lpm3. The password is case-sensitive. **Make sure you remember what your new password is – it may take several days to recreate your account if you forget your password! If you do forget your password, you will need to send an email to** [**F.Mwagwabi@murdoch.edu.au**](mailto:F.Mwagwabi@murdoch.edu.au) **asking very nicely to have your password reset.**

You are now logged in to Oracle and should see the SQLPlus prompt **SQL>** as shown in the screenshot below. This means you can enter SQL commands.

Text

Description automatically generated

## Changing your password again

If you want to change your password at any other time, at the SQL prompt, enter **password;** and follow the instructions.

**Logging out of sqlplus and rhea.ad.murdoch.edu.au**

* **To exit from Oracle, type exit**
* **And then to exit from rhea.ad.murdoch.edu.au type exit**

**The preferred method** for you to log into Oracle SQLplus (once you have changed your password) is to **use the Oracle SQL Developer GUI Client**.

# Logging on to Oracle SQLplus through the SQL Developer GUI client

SQL Developer is a GUI client that allows the user to interact with the Oracle database in a slightly more friendly way than via a simple command line. It is available free from Oracle and can be downloaded from: <http://www.oracle.com/technology/products/database/sql_developer/index.html>

You may need to sign up for a free Oracle account to download the file.

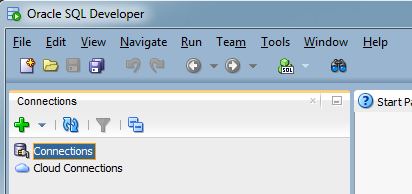
It available for several different platforms and the installation is relatively straightforward.

**It is important to note that you must still log in to Oracle using the SSH client the FIRST TIME you log in to Oracle in order to unlock your account (or any time that your password has been reset). This cannot be done using Oracle SQL Developer.**

* Log on to your computer in the usual way.
* In the on-campus computer labs at the Murdoch Campus, Choose Start-Programs, then Oracle SQL Developer.
* If you have installed SQL Developer at home, start it by selecting the SQL Developer executable.

**In order to connect to the database, you need to create a new Database Connection:**

* Click the New Connection button 



New Connection

Figure 1: Oracle SQL Developer Initial Screen

* Enter the following details into the New/Select Database Connection dialogue

|  |  |
| --- | --- |
| Connection Name: | STUDENT |
| Username: | Your username is an uppercase A followed by your student number. For example, if your student number is **20012001**, your username and initial password would be **A20012001** |
| Password: | Your new password (see section 1 above) |
| Role: | Select “default” |
| Connection Type: | Select “Basic” |
| Hostname: | rhea.ad.murdoch.edu.au |
| Port: | 1580 |
| SID: | Student |

Graphical user interface, text, application, email

Description automatically generated

Figure 2: New/Select Database Connection Dialogue

* Click Save so that you don’t have to re-enter the information on subsequent logins. NB: Your password will not be saved and you will need to enter it for future logins.
* Click Connect.

## ROLLUP, CUBE, GROUP

**In this part of the lab**, you will create a dataset consisting of a **Fact table** populated with random data. There will be Foreign Key values, but **we will not create the Dimension table**s for this exercise.

The Fact table we will create and use is called **tbl\_Fact**. It consists of:

1. four foreign keys (**dim\_1\_fk**, **dim\_2\_fk**, **dim\_3\_fk**, **dim\_4\_fk**) and,
2. a value (measure) called **sale\_total** which represents the total value of a sale.

🡪 If you have already created tbl\_Fact, you will need to delete it. Otherwise, create the table using the following SQL:

CREATE TABLE tbl\_Fact (

sale\_ID NUMBER GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),

dim\_1\_id NUMBER NOT NULL,

dim\_2\_id NUMBER NOT NULL,

dim\_3\_id NUMBER NOT NULL,

dim\_4\_id NUMBER NOT NULL,

sale\_total NUMBER(10,2) NOT NULL,

CONSTRAINT fact\_pk PRIMARY KEY (sale\_ID));

🡪 You will now create 1000 rows of random data to populate the table for the rest of the exercise:

INSERT INTO tbl\_Fact (dim\_1\_id, dim\_2\_id, dim\_3\_id, dim\_4\_id, sale\_total)

SELECT

TRUNC(DBMS\_RANDOM.value(low => 1, high =>3)) AS dim\_1\_id,

TRUNC(DBMS\_RANDOM.value(low =>1, high =>6)) AS dim\_2\_id,

TRUNC(DBMS\_RANDOM.value(low =>1, high =>11)) AS dim\_3\_id,

TRUNC(DBMS\_RANDOM.value(Low =>1, high =>11)) AS dim\_4\_id,

ROUND(DBMS\_RANDOM.value(low =>1, high =>100),2) AS sale\_total

FROM dual

CONNECT BY level <= 1000;

🡪 You should now COMMIT those inserts!

### GROUP BY

While I am sure you remember from your previous database studies how GROUP BY works, let’s just have a quick reminder ☺

🡪 Show the total value of all of sales in your dimension table, you would use the following:

SELECT SUM(sale\_total) as TOTAL\_SALES

FROM tbl\_fact;

TOTAL\_SALES

-----------

49923.48

\*\*\*Your value will differ to this as the values are random.

The **column ~~fact~~ dim\_1\_id** has only two distinct values, 1 and 2 (you could, of course check this, by using SELECT DISTINCT!) To see the value of sales of these two values, you would use the GROUP BY operator:

SELECT dim\_1\_id, COUNT(\*) as num\_rows, SUM(sale\_total) as TOTAL\_SALES

FROM tbl\_fact

GROUP BY dim\_1\_id;

DIM\_1\_ID NUM\_ROWS TOTAL\_SALES

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

1 504 25118.66

2 496 24804.82

\*\*\*Your values will differ.

🡪 What would happen if you didn’t include the GROUP BY statement in this query? Try it and see ☺

🡪 Write a query to determine the number of rows and total sales grouped by both dim\_id\_1 and dim\_id\_2. \*\*\*HINT: there should be 10 rows in your result.

🡪 Write a query to determine the number of rows and total sales grouped by dim\_id\_1, dim\_id\_2 and dim\_id\_3. \*\*\*HINT: there should be 100 rows in your result.

### ROLLUP

The **ROLLUP command** is used with GROUP BY, and will **produce a sub-total** for each grouping, as well as a grand-total for the query.

This query is answering the question, “Provide a query that SUMS the sales for the combinations of dim\_1\_id and dim\_2\_id, and a subtotal for each value of dim\_1\_id, and the total sum of all sales.”

SELECT dim\_1\_id, dim\_2\_id, SUM(sale\_total) as TOTAL\_SALES FROM tbl\_fact

GROUP BY ROLLUP (dim\_1\_id, dim\_2\_id)

ORDER BY dim\_1\_id, dim\_2\_id;

If you think about what the query is doing, and look at the results of the query, you can see it is doing the following:

* **Grouping the rows** by the various combinations of the values of dim\_1\_id and dim\_2\_id, where dim\_1\_id = 1. That means there will be a group where dim\_1\_id = 1 and dim\_2\_id =1, and then another where dim\_1\_id = 1 and dim\_2\_id =2, and so on…
* **Calculating the sum** of the sale\_total value for each of those groups
* When all of the combinations where dim\_1\_id =1 are calculated, it is then calculating the SUM of all of those groups.
* The process will be repeated for all rows where dim\_1\_id =2.
* When all groups have been completed, the final calculation is the SUM of all values.

DIM\_1\_ID DIM\_2\_ID TOTAL\_SALES

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

1 1 4648.47

1 2 4298.21

1 3 5959.34

1 4 4860.17

1 5 3803.5

1 23569.69

2 1 5098.12

2 2 5795.55

2 3 4583.17

2 4 4826.27

2 5 5002.55

2 25305.66

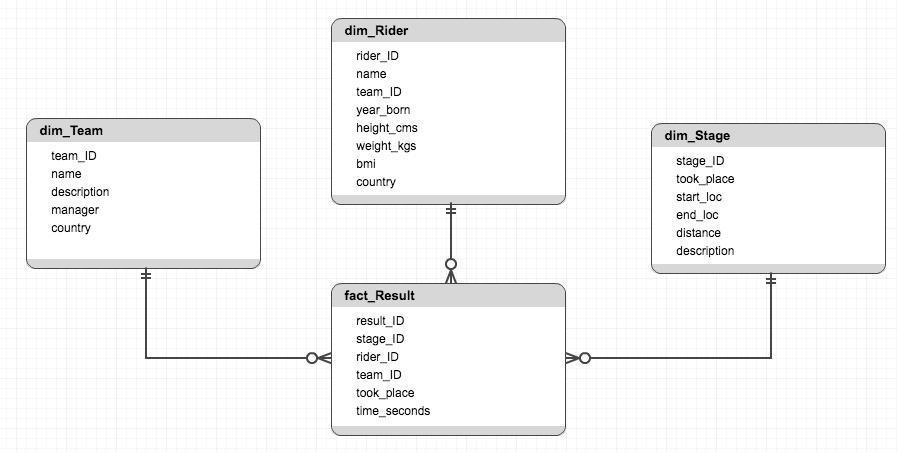
48875.35

13 rows selected.

🡪 Try changing the order of the columns in the GROUP BY ROLLUP, and ORDER BY to see how the results are effected.

### Denormalisation

Below is a small data warehouse of the 2012 Tour de France as detailed in the star schema below:



🡪 The star schema above contains several examples of denormalization. What are they and why do you think the decision to denormalize may have been made?

1. If you are going to access the server in this way, you would use the command – “ssh –l YOUR\_STUDENT\_NUMBER rhea.ad.murdoch.edu.au” in a terminal session [↑](#footnote-ref-1)
2. For a fuller list of SSH clients, see: <http://en.wikipedia.org/wiki/Comparison_of_SSH_clients> [↑](#footnote-ref-2)