Using the Oracle SQL Developer Tool

Contents

[1 Development Tools 1](#_Toc534192552)

[1.1 Oracle SQL Developer 1](#_Toc534192553)

[1.2 Registering for Oracle 2](#_Toc534192554)

[1.3 Start SQL Developer 2](#_Toc534192555)

[1.4 Creating a Connection 3](#_Toc534192556)

[2 Using the Developer Tool 5](#_Toc534192557)

[2.1 Your Database Objects 5](#_Toc534192558)

[2.2 Running a query 5](#_Toc534192559)

[2.3 Importing Data 6](#_Toc534192560)

[2.3.1 Step 1 Data Preview 6](#_Toc534192561)

[2.3.2 Step 2 Import Method 8](#_Toc534192562)

[2.3.3 Step 3 Choose Columns 8](#_Toc534192563)

[2.3.4 Step 4 Column Definition 8](#_Toc534192564)

[2.4 Region Table 10](#_Toc534192565)

[3 Import Data 11](#_Toc534192566)

[3.1 Population Table 13](#_Toc534192567)

[3.2 Joining the tables 14](#_Toc534192568)

[4 Exiting Developer 16](#_Toc534192569)

[5 Summary 16](#_Toc534192570)

# Development Tools

You will use a variety of tools throughout this module to handle big data, including Oracle, MongoDB and Hadoop. This workbook will look at the Oracle SQL Developer tool and how it can be used to import external data.

## Oracle SQL Developer

For home use, the Oracle SQL Developer tool is also available free to students from here:

<http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html>

If you select a version without the JDK, you must already have JDK 8 or above installed on your machine. Note, it must be a JDK version of Java, not JRE. If you are not sure what type of Java you have installed, use a version with the JDK included.

Oracle SQL Developer is a graphical tool for database development.

You can:

* browse database objects
* run SQL statements and SQL scripts
* edit and debug PL/SQL statements.
* create and run reports
* view Data Modeler Diagrams
* connect to the database
* import and export data

The aim is to simplify database development tasks and aid productivity.

Note the version currently installed in the University labs will not always be the latest version, so you might find if you download it for home use, the layout of the window settings can vary slightly from version to version.

## Registering for Oracle

If you do not have an Oracle account please go to this page first to register:

<https://mi-linux.wlv.ac.uk/~oracle/reg12c/orareg.php>

Do note, even if you had an account in your second year, it may have been cleared out at the end of the academic year.

Useful Oracle links:

|  |  |
| --- | --- |
| Using Oracle at the University | <http://oradb-srv.wlv.ac.uk/info/Oracle.html> |
| Oracle Wiki | <https://mi-linux.wlv.ac.uk/wiki/index.php/Oracle> |

For the purposes of this module we will be using the Developer tool, rather than SQL\*Plus.

## Start SQL Developer

Start Oracle SQL Developer, which can be found with the other Oracle tools found on Apps Anywhere (search for Oracle). The first time you run it, you will get something similar to the following window (exact version number may vary):

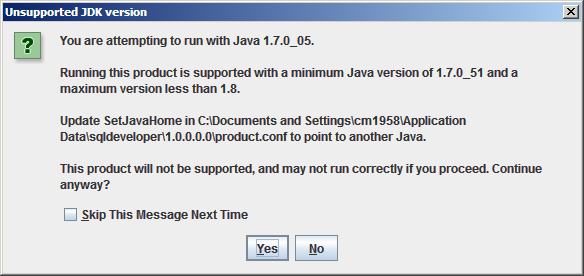


Figure ‑ Java 1.7 Warning

Click on Yes to continue and to avoid getting the message again also click the Skip This Message Next Time box.

You can also say No to the following:

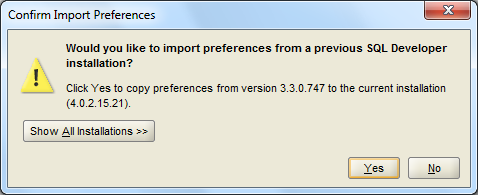


Figure ‑ Import Preferences

And it is purely up to you if you want to allow Oracle to track usage:

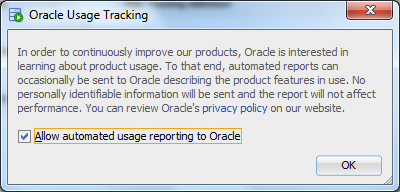
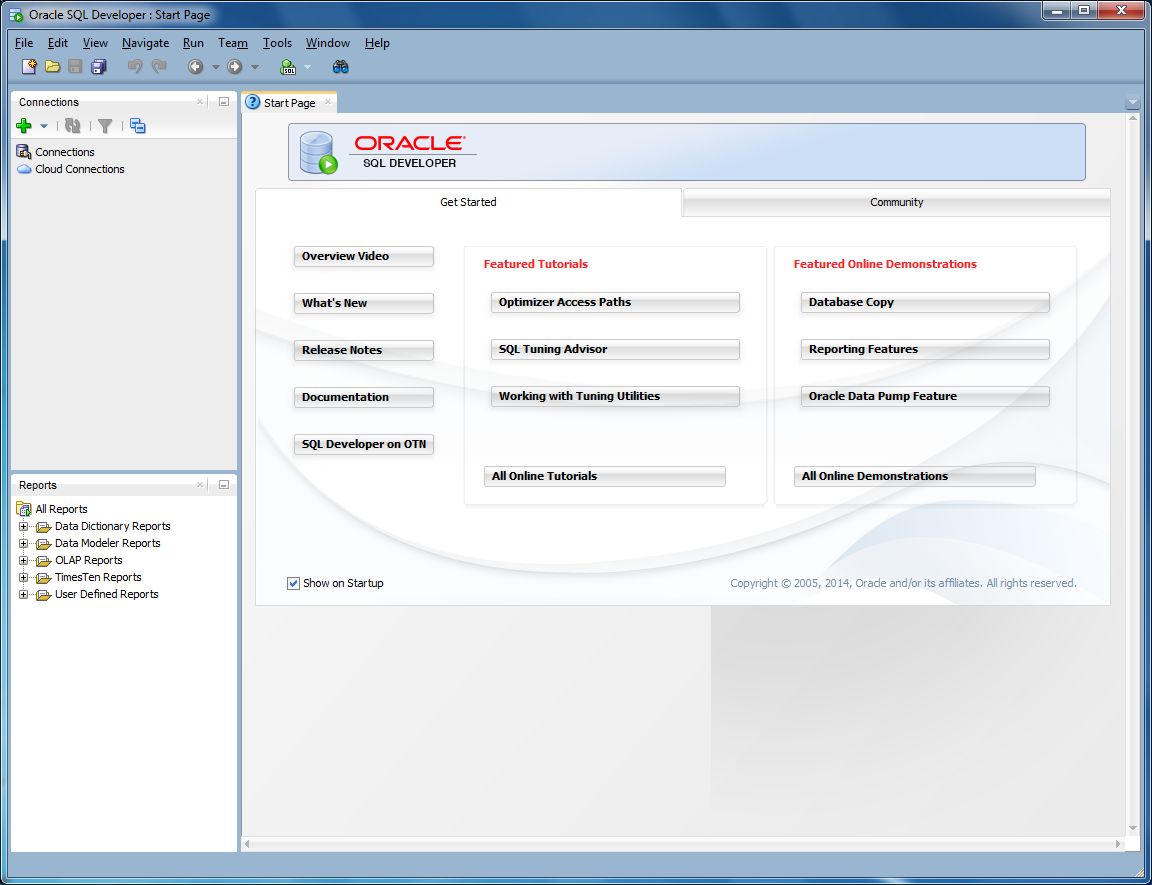


Figure ‑ Oracle Usage Tracking

Unlike the Data Modeler Tool, you can connect to a database from here, so the first thing to do is create a connection (figure 1-4):



Click on the green cross to setup a connection

Figure ‑ Initial SQL Developer Screen

## Creating a Connection

To login to Oracle you need to setup the following details:

Connection Name: this can be called anything you like, e.g., myOracle

Username: Your Oracle Username, usually ops$studentNo

Password: Your Oracle password

Click on Save Password if you wish to keep the password for future use.

Connection Type: Basic

Hostname: address for the Oracle Server – oradb-srv.wlv.ac.uk

Port: portno of the Oracle database server – 1522

SID: Name of the Oracle database – orcle12c

The final settings should be similar to figure 1-5:

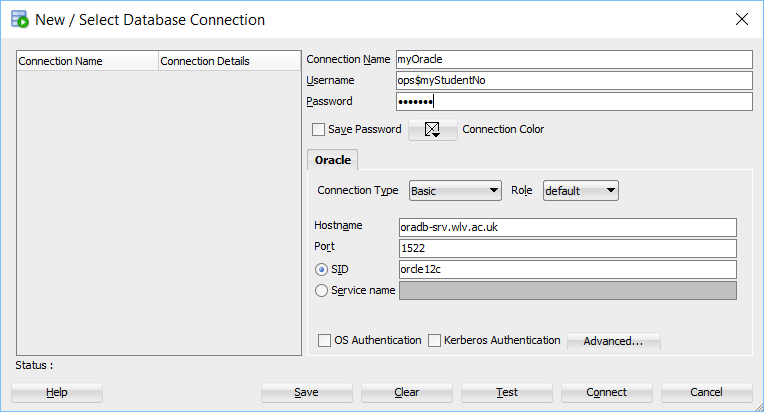


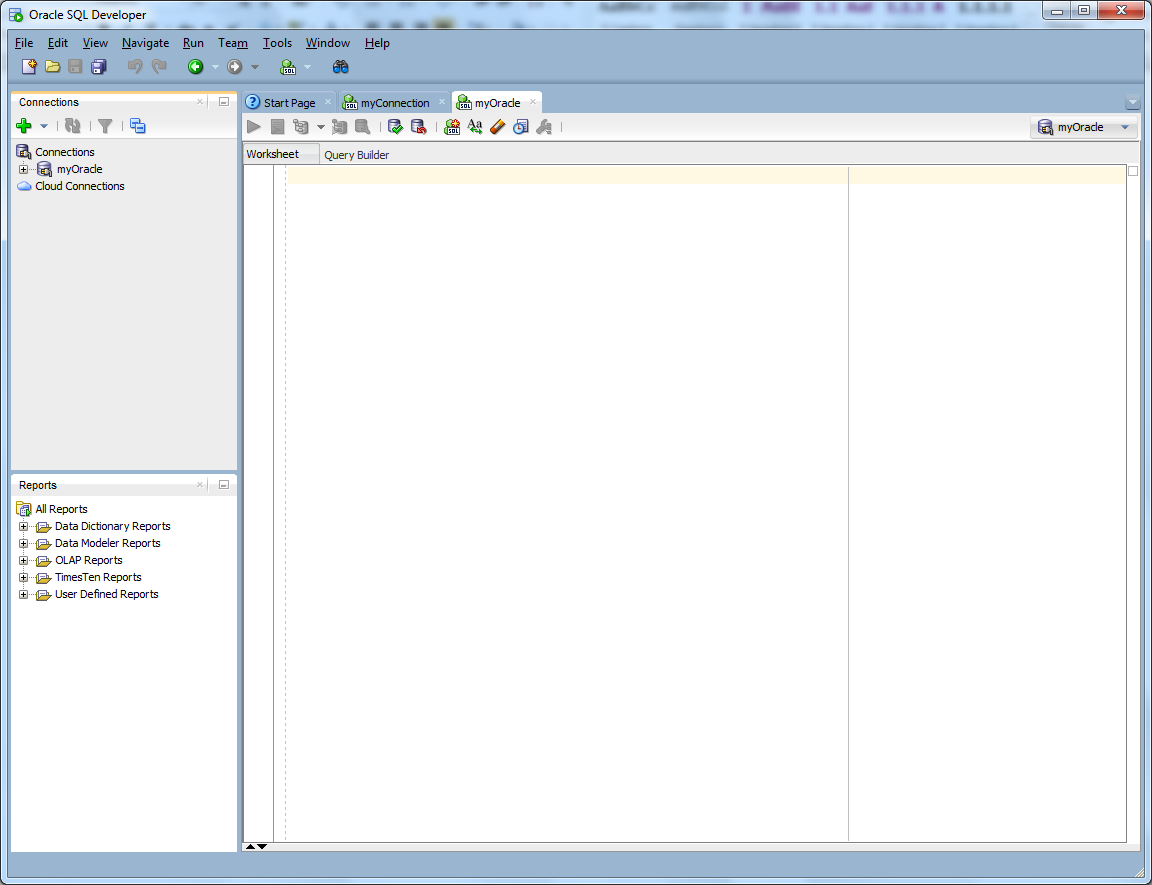
Figure ‑ Database Connection Window

* Note, these details can be used from any computer in the University, or from your home PC

Click on Test to check the details are correct and if it reports the *Status: Success*, then click on Save to keep the details and Connect to login.

If logged in successfully, the initial screen will be similar to Figure 1-6.

In the Worksheet area, you can directly write SQL code, in the same way you would have used the SQL\*Plus interface.



Worksheet – SQL queries can be typed in this window

Figure ‑ Initial Developer Screen

# Using the Developer Tool

## Your Database Objects

You can view what tables you have created, or any other objects, such as triggers or views. If you have logged in successfully, click on the + symbol beside your connection name: . You will then see all the various objects in your database:

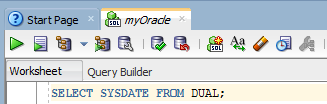
|  |  |
| --- | --- |
| Figure ‑ Database Objects | Click on the  symbol against any of these to see what objects you own. |

## Running a query

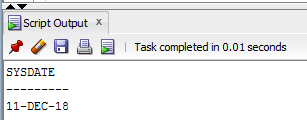
In the Worksheet area seen in Figure 1-6 type in the following:

SELECT SYSDATE FROM DUAL;

How you run a query in SQL Developer is different to using SQL\*Plus, you cannot simply press return. At the top of the window is a set of options:



Click on the first large Green button, which is the Run Statement button (or press Ctrl+Enter). Check the Script Output Window in the lower pane for the results:



The second green button is the Run Script button, which can be used if you want to run more than one query at the same time, e.g., if you have a script file with several SQL statements in it.

Note,

* If you are prompted for a Database connection – use the name of the connection you setup in Section 1.4
* If you inadvertently close the worksheet by mistake, right-click on the name of your connection and select Open SQL Worksheet

## Importing Data

The workspace can be used to type in any command that you would use in SQL\*Plus. One thing you cannot easily do in SQL\*Plus is import data. This module is about manipulating data from external sources, so we will look at importing an Excel spreadsheet, since this is often a method used to transfer data from one system to another.

You can either define the table beforehand, or get Oracle to use the first line of the file as table headings, which we will do:

1. Download the *Population Estimates (aged 16-64)* spreadsheet from Canvas onto your computer. This is an example of local authority data.
2. This file is small enough to be viewed using Excel, but the purpose of this exercise is to understand how external files can be imported into a database. This method will also work with Comma Separate Value (CSV) files.
3. Under your Connection Name (e.g., *myOracle*) right-click on the Tables (Filtered) option and select Import Data…. (Figure 2-2):

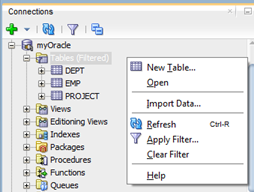


Figure ‑ Import Data Command

The Import Wizard will then take you through a set of steps:

### Step 1 Data Preview

First select the spreadsheet in the Import Data File: box. Use Browse to navigate to where you saved the spreadsheet.

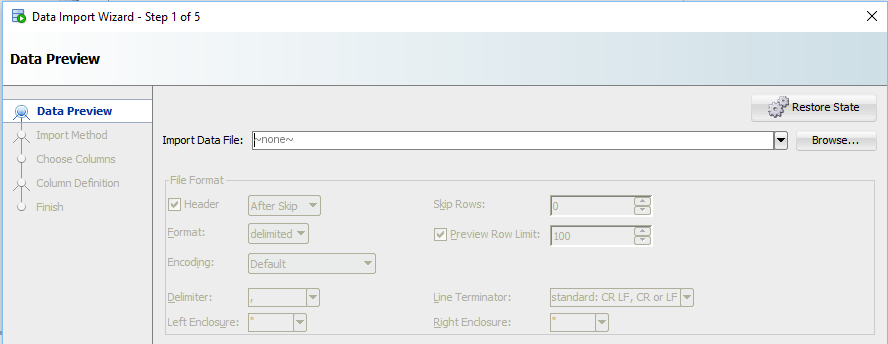


Figure ‑ Import Step 1

For example:

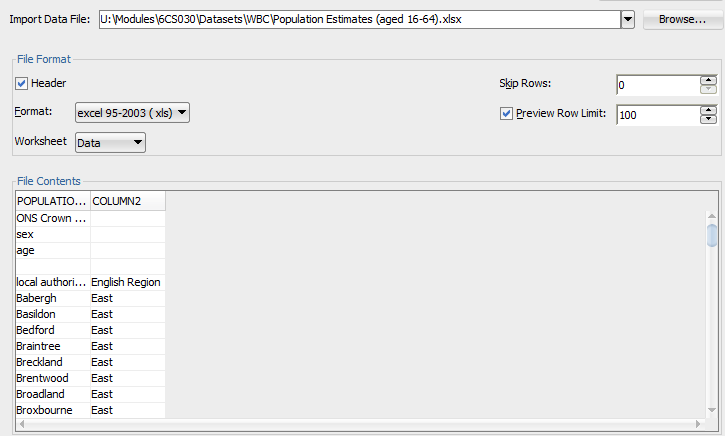


Figure ‑ Import Data: Step 1

The options that were greyed out before now become active:

* Header: the tick means the first row includes the column headings
* Format: is the file format. This file is a newer version of Excel, so click on the arrow to select 2003+ (xlsx)
* Worksheet: this spreadsheet has 2 Worksheets, so click on the Worksheet button and change the value to Region List.
* Skip rows: leave this at 0. This can be used this if you want to skip a number of records, for example, if the Preview option had been used to load a sample set of data and you later wanted to add the rest of the data.
* Preview Row Limit: can be used to load a sample of data. It defaults to 100. Untick the box to load all the data:

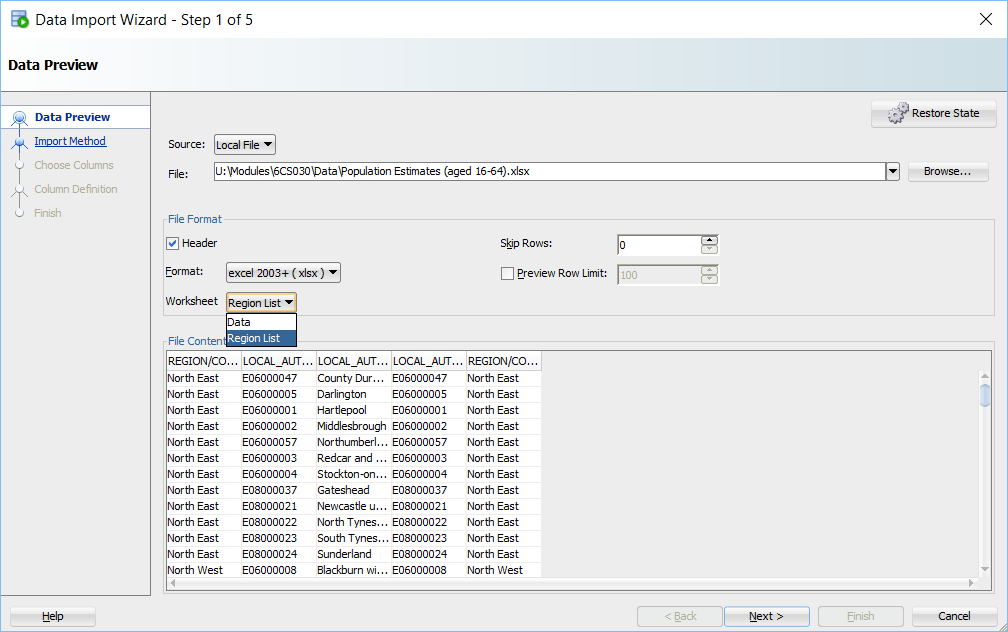


Figure ‑ Final Step 1 Options

When finished, click on Next to continue.

### Step 2 Import Method

This step asks for a table name, call it: REGION and press Next again.

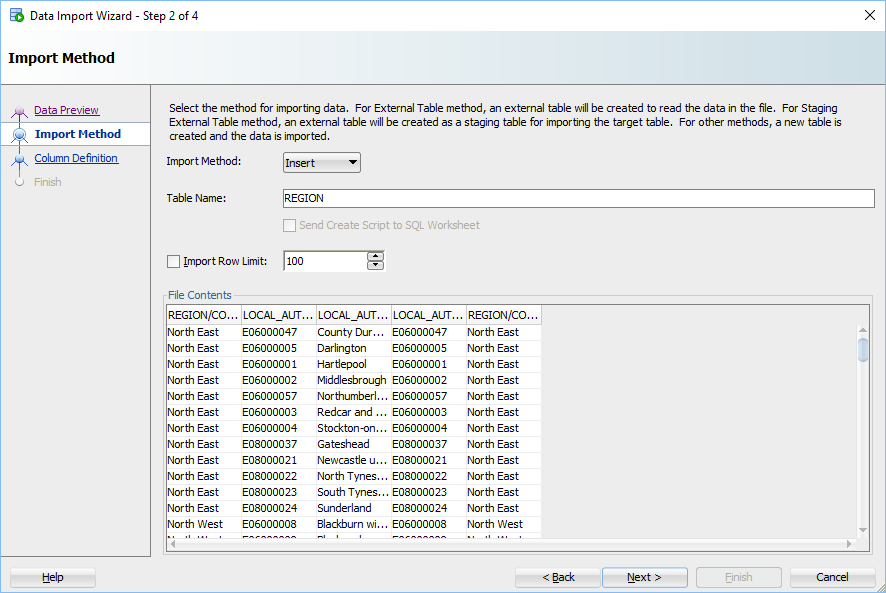


Figure ‑ Step 2

Note, on this screen you can limit the number of rows to be imported (Import Row Limit) again. Leave unticked, but could be a useful option if you just wanted to get a flavour of the data in a large dataset.

### Step 3 Choose Columns

The next step lets you pick which columns you want to import. In this case we want them all selected, but can be used to import just a subset of the data. Press Next again:

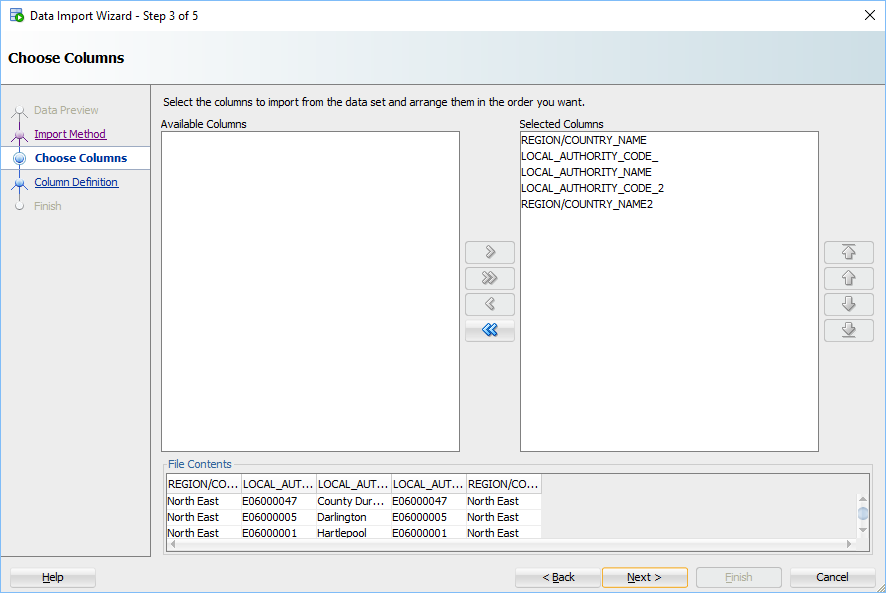


Figure ‑ Step 3 Choosing Columns

### Step 4 Column Definition

The next step checks the column definitions. The red exclamation marks means some changes are needed before continuing:

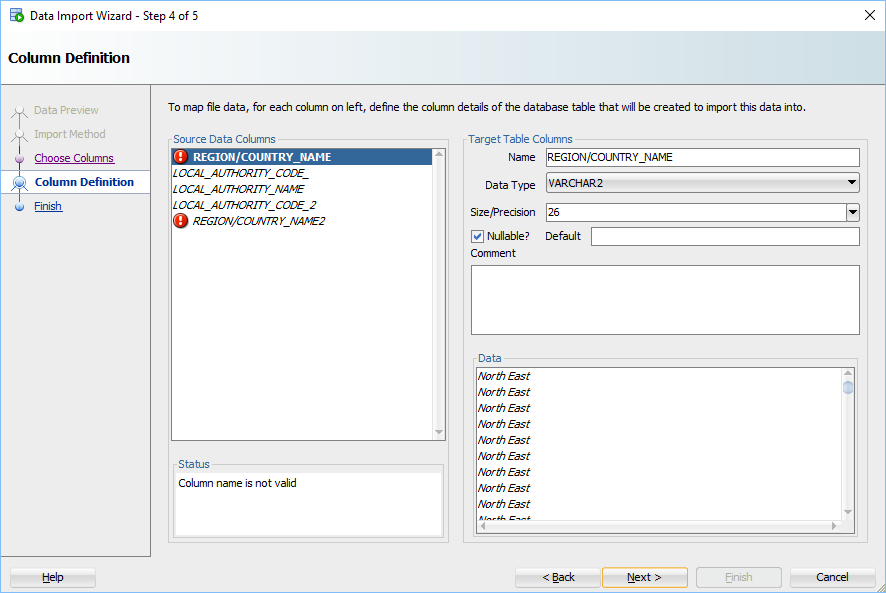


Figure ‑ Column Names - Before

The problem is that Oracle does not allow “/” in the field names, unless you enclose the name in double quotes. This makes any SQL commands cumbersome, so it is easier to remove them.

COUNTRY is probably a typo for COUNTY, so change these two column names to: COUNTY\_NAME and COUNTY\_NAME2

Also remove the trailing underscore from: LOCAL\_AUTHORITY\_CODE\_ to become: LOCAL\_AUTHORITY\_CODE

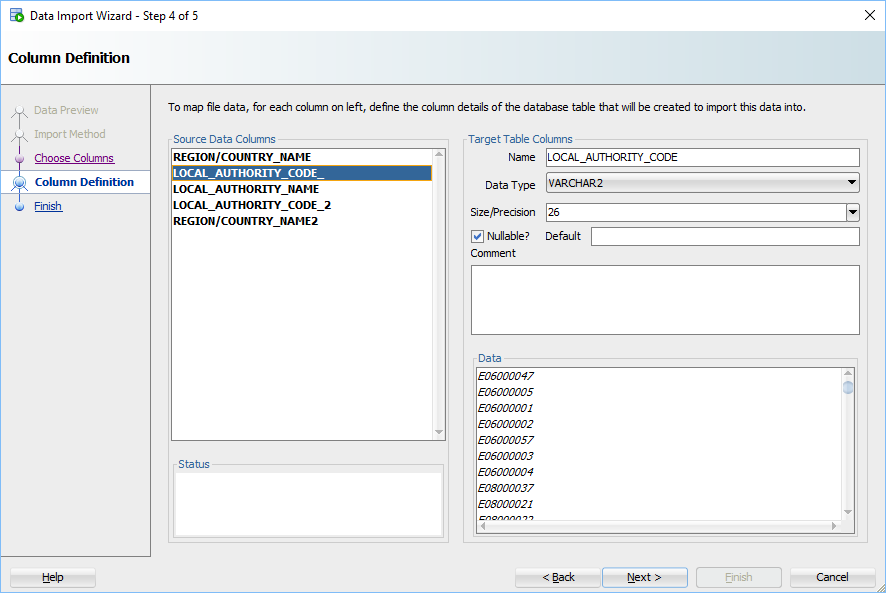


Figure ‑ Colum Names - After

Oracle will also have examined the data to determine the data type and what size the fields should be. E.g., LOCAL\_AUTHORITY\_CODE in Figure 2-9 is set to a VARCHAR2 and size 26. This could be amended if you think any future data might need a bigger size.

The final step allows you to verify if everything is ok before importing:

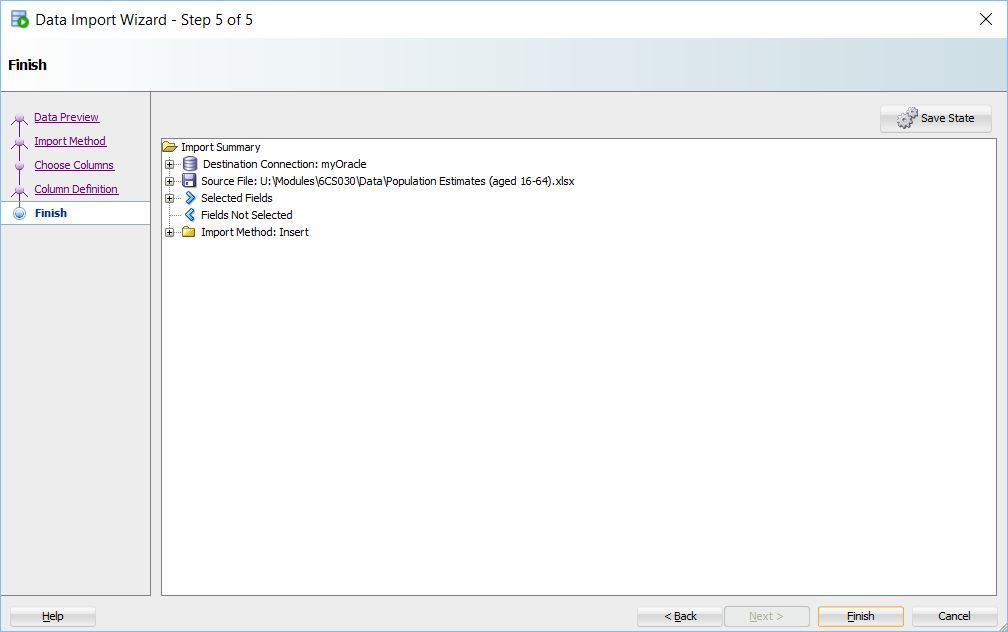
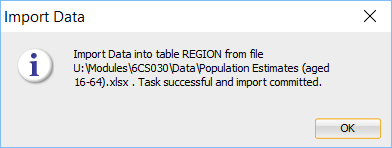


Figure ‑ Final Step

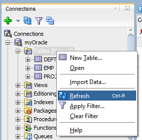
If not, the <Back button can be pressed so you can check to see if something was not setup correctly in a previous step.

Otherwise press the Finish button and the system should reply that the data was imported successfully:



Note, with the Import Data function the data is committed automatically.

## Region Table

To check that the table has been imported successfully, right click on the Tables (Filtered) option and select Refresh: 

The Region table should now exist. If you click on Region the table structure can be seen in the right-hand pane (figure 2.11):

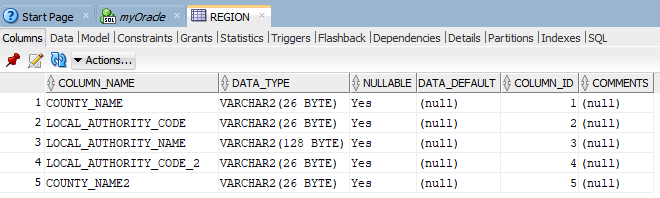
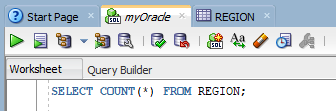


Figure ‑ Region Table

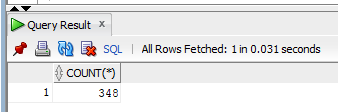
To view the values, click on the Data tab that can be seen along the top of this pane.

If you want to run SQL commands, click on the tab that has the same name as your connection, for example: myOracle as seen above. This will open up a Worksheet where you type in SQL commands.

To check how many rows were imported into the Region table, type in the following SQL command:



Pressing return will not run the command as seen in SQL\*Plus. Remember to press the large green button and a new Query Result pane will appear at the bottom of the window:



If you check the spreadsheet in Excel, you will find there were indeed 348 rows of data.

Try some more commands to get a feel for the data:

SELECT COUNTY\_NAME, COUNT(\*) AS COUNT FROM Region

GROUP BY COUNTY\_NAME;

SELECT \* FROM Region

WHERE COUNTY\_NAME = 'West Midlands'

ORDER BY LOCAL\_AUTHORITY\_NAME;

# Import Data

For this next section we will repeat the same exercise with the Data worksheet that can also be found in this spreadsheet.

Before importing the data, open the file using Excel. Have a look at the data.

What issues might the following pose?

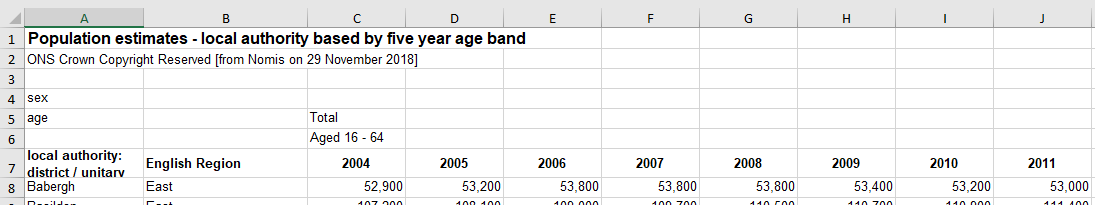
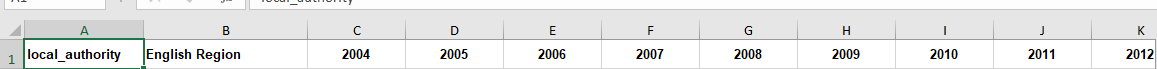


Figure ‑ Data Workbook

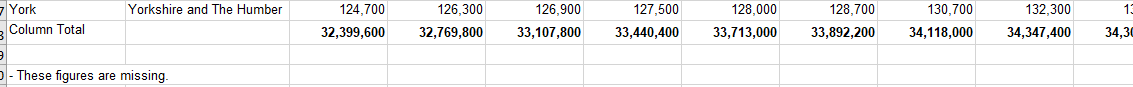
If you remember Oracle (and most other systems that import data), expect the first row to contain the column headings. This clearly does not, so before importing delete lines 1-6 so that line 7 becomes the first line.

The first column heading name: *local authority: district / unitary (prior to April 2015)* is going to cause problems when importing since the length is too long for a column name, so rename it to: local\_authority

The first line should now look like this:



Also look at the bottom of the file:



We don’t need the Column Total record; which can be deleted. These figures can be generated using SQL, however, before removing make a note of the 2016 and 2017 figures:



We also do not need the line stating “*- These figures are missing”* either, so remove this line too. The last line should be the data for York.

Save the Excel file with a new name, for example: Population Estimates Updated. This means that you can refer to the original if necessary.

Then follow the steps from Section 2 to load this workbook into Oracle. Call this table POPULATION

At Step 4 you fill find Oracle does not like all the year columns. This is because column names cannot begin with a number. Note, this is true for all Oracle objects:

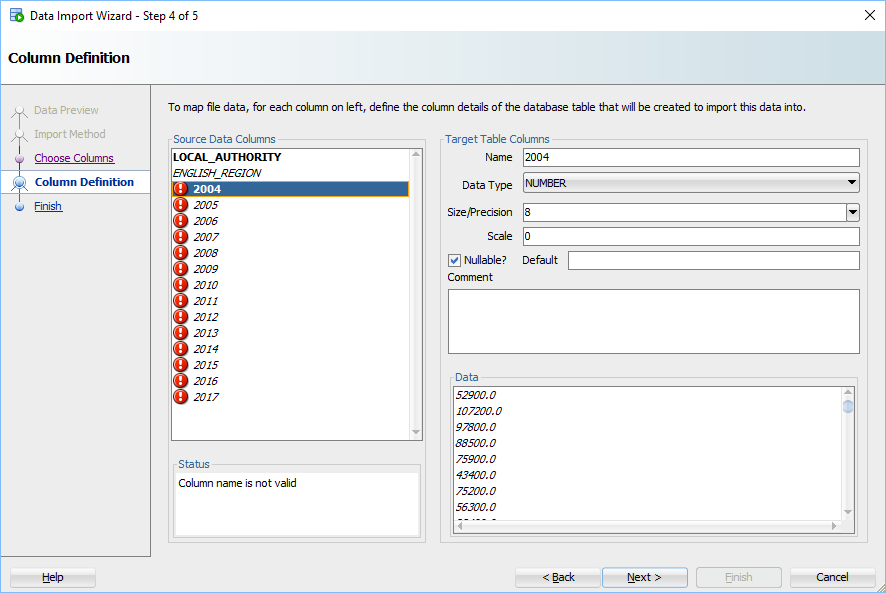


Figure ‑ Year Column Definitions - Before

Prefix each of the year columns with LA\_

Whilst you are doing this also check that the Datatype for each year column is a Number. The Size/Precision is usually 8, which should be big enough:

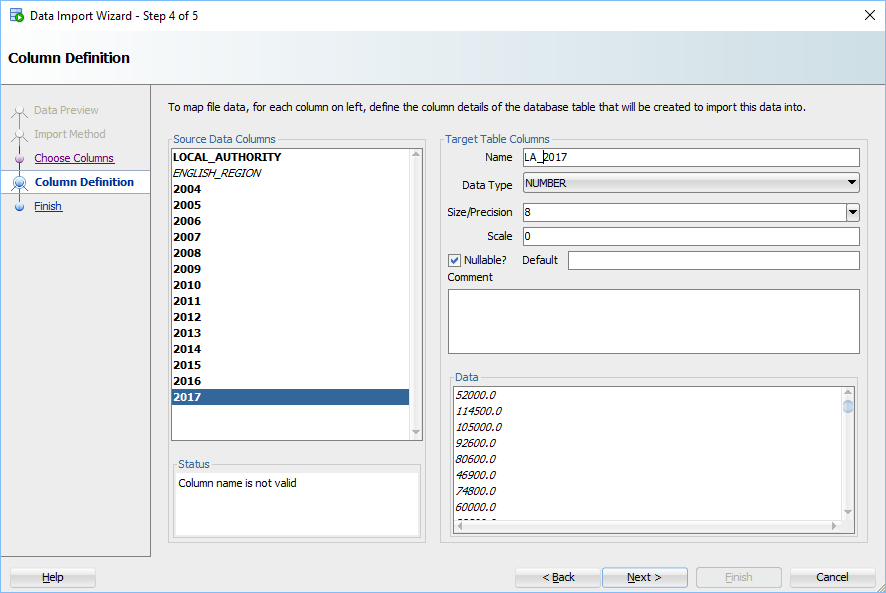


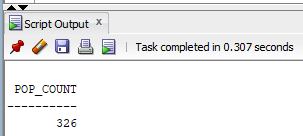
Figure ‑ Year Column Definitions – After

## Population Table

Refresh the Tables list to see if the new table is there and try running some commands:

**SELECT COUNT(\*) AS POP\_COUNT FROM POPULATION;**

The system should respond with:



Generate the column totals for 2016 and 2017:

SELECT SUM(LA\_2016) AS "2016", SUM(LA\_2017) AS "2017"

FROM POPULATION;

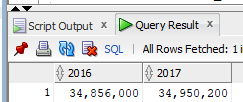
To format the numbers to include 1000 separators, use TO\_CHAR:

SELECT TO\_CHAR(SUM(LA\_2016),'999,999,999') AS "2016",

TO\_CHAR(SUM(LA\_2017),'999,999,999') AS "2017"

FROM POPULATION;

Note, did these totals match the Column totals removed earlier:



If not, try using the SUM function in Excel to see if they agree with Oracle’s figures. This is something we will come back to when looking at cleaning found data.

## Joining the tables

When carrying out statistical work, you may need to join the tables imported, but this must be done sensibility to produce meaningful results. So which columns can be used for the join? If you view the data of both tables, the LOCAL\_AUTHORITY column in Population seems to match the LOCAL\_AUTHORITY\_NAME in Region and ENGLISH\_REGION has similar data to COUNTY\_NAME in Region:

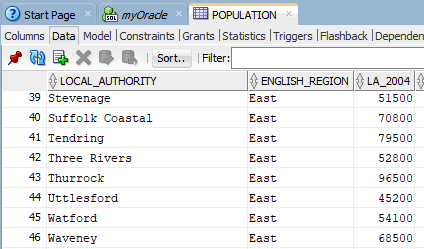


Figure ‑ Population Table

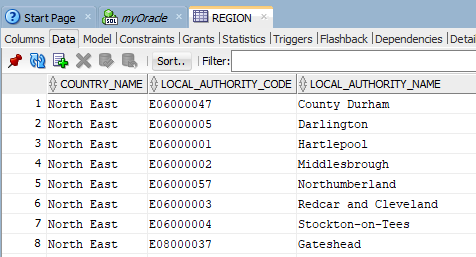


Figure ‑ Region Table

The regional names have duplicate values, but from a first glance the local authority names seem to be unique within each table. One way to check this is to use DISTINCT to remove any potential duplicate values and see if the total match the table counts from above:

SELECT COUNT(DISTINCT LOCAL\_AUTHORITY) FROM POPULATION;

SELECT COUNT(DISTINCT LOCAL\_AUTHORITY\_NAME ) FROM REGIONS;

Both appear to be a match, so if we want to join both tables the local authority name seems ideal for the join condition.

Do note, this does not necessarily mean all authorities have been covered in both tables. An issue to consider when looking at cleaning the data next week.

To see some details from both tables, joined over the local authority name:

SELECT COUNTY\_NAME, LOCAL\_AUTHORITY\_NAME, LOCAL\_AUTHORITY\_CODE,

LA\_2015, LA\_2016, LA\_2017

FROM REGION R, POPULATION P

WHERE R.LOCAL\_AUTHORITY\_NAME =P.LOCAL\_AUTHORITY

ORDER BY COUNTY\_NAME, LOCAL\_AUTHORITY\_NAME;

# Exiting Developer

Before leaving Developer disconnect from the database by right-clicking on your connection name under the Oracle Connections pane, then select Disconnect:

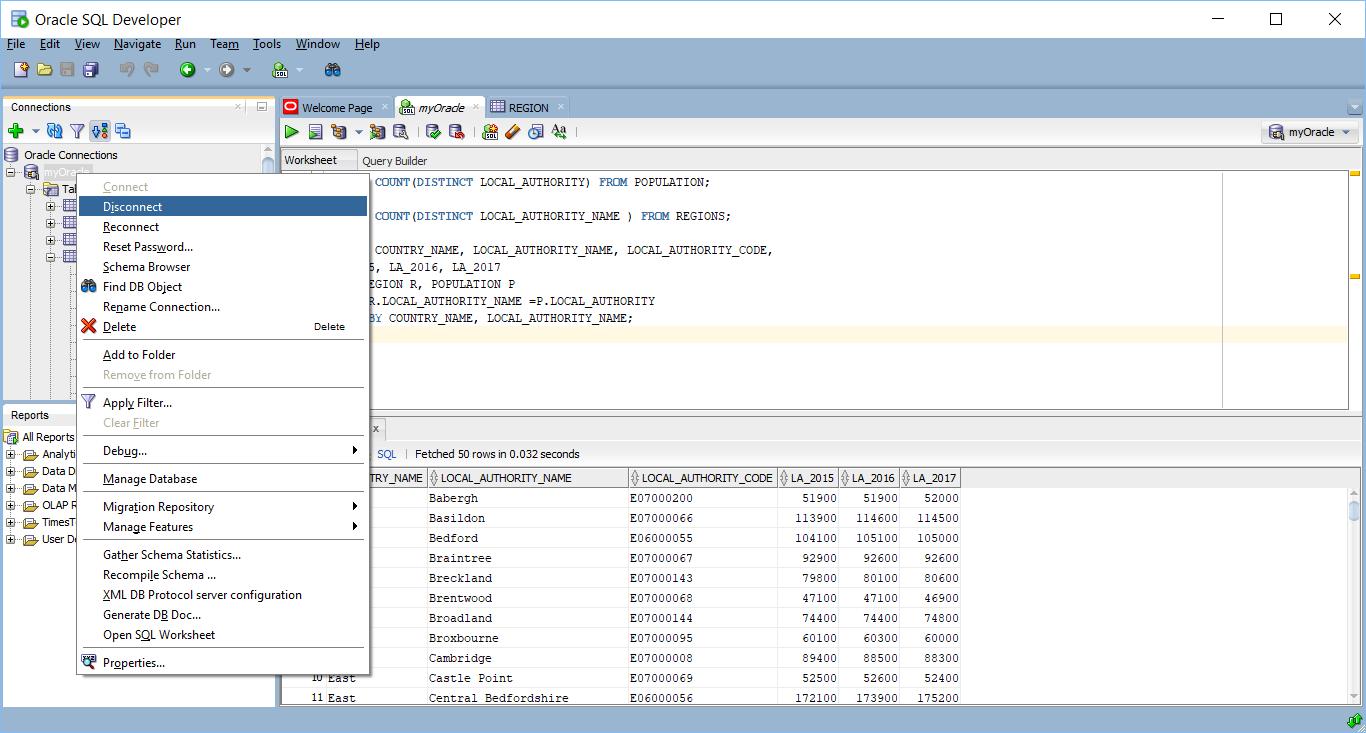
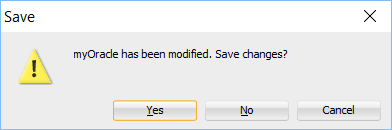


Figure ‑ Disconnecting From Oracle

Then click on File>Exit to leave. You will be prompted to save the queries from your worksheet:



Press Yes if you wish to save the queries, otherwise press No.

# Summary

The Developer Tool is quite sophisticated and can be used for creating and maintaining other database objects.

If you are interested in finding out more about the Developer tool, the online documentation has further details:

* <http://oradb-srv.wlv.ac.uk/E50529_01/index.htm>
* <http://www.oracle.com/technetwork/developer-tools/sql-developer/documentation/index.html>