

# CS992: Database Development – Lab 1

## Using Oracle's *Live SQL* platform

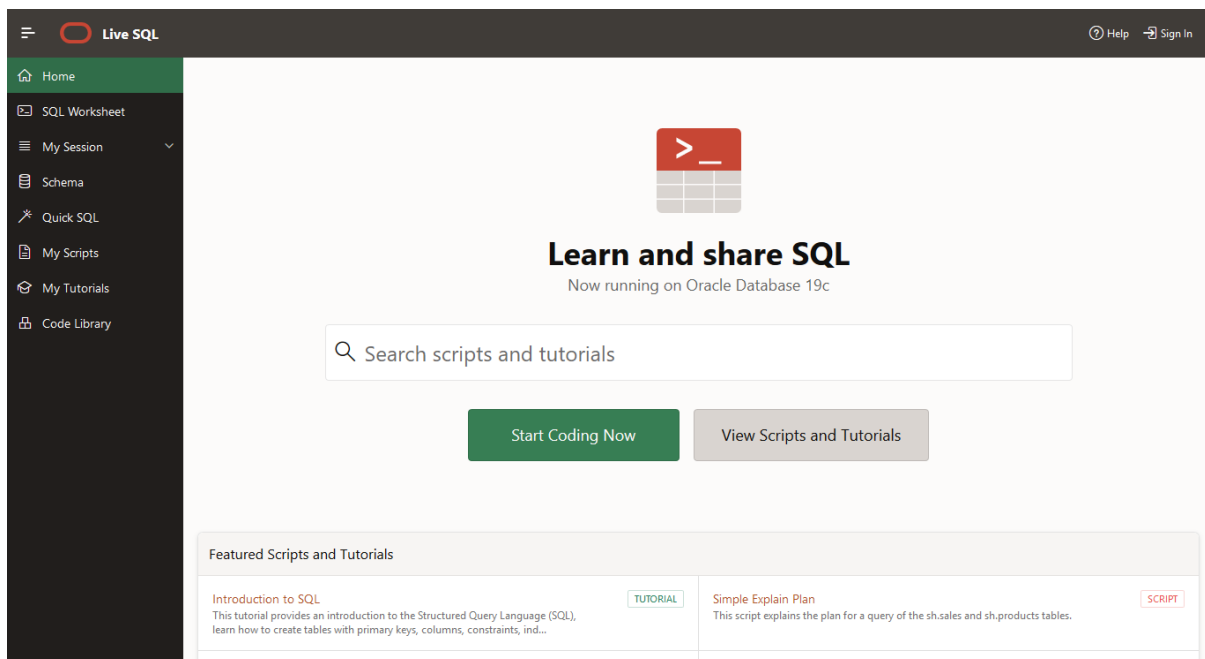
You have already been exposed to an approach that involved running Oracle on the local CIS departmental server, using a terminal window in command-line mode. From experience, we have found that trying to execute PL/SQL in various forms (as we will be doing in Labs 2 and 3) can be problematic in that context. As such, we have chosen to use Oracle's *Live SQL* platform to run the set of labs linked to this module. Apart from anything else, this also gives you exposure to a different database environment and there are many useful resources available within the *Live SQL* framework. In common with other DBMS providers, Oracle is continually expanding the number and range of services offered via the *Cloud*. You can read more about this by taking a look at their, "What's New in Application Development" page -

<https://www.oracle.com/database/technologies/application-development.html>

In the context of our lab activities, we will be using the *Live SQL* platform – “a free, browser based tool, that you can use to write SQL”. You can click on the *Live SQL* link shown on the Oracle page above, or go to the platform directly at -

<https://livesql.oracle.com/>

Either way, you should end up at a page that looks something like this... (you will need to click on the little navigate icon at top-left to bring up the left-hand panel).



You should see a “Sign In” icon at the top-right of the page. While the use of *Live SQL* is free, you do need to register with Oracle for an account (assuming that you do not already have one). This is a painless process and will take about a minute – after you have done this you will be ready to get started.

## 1. Getting familiar with *Live SQL*

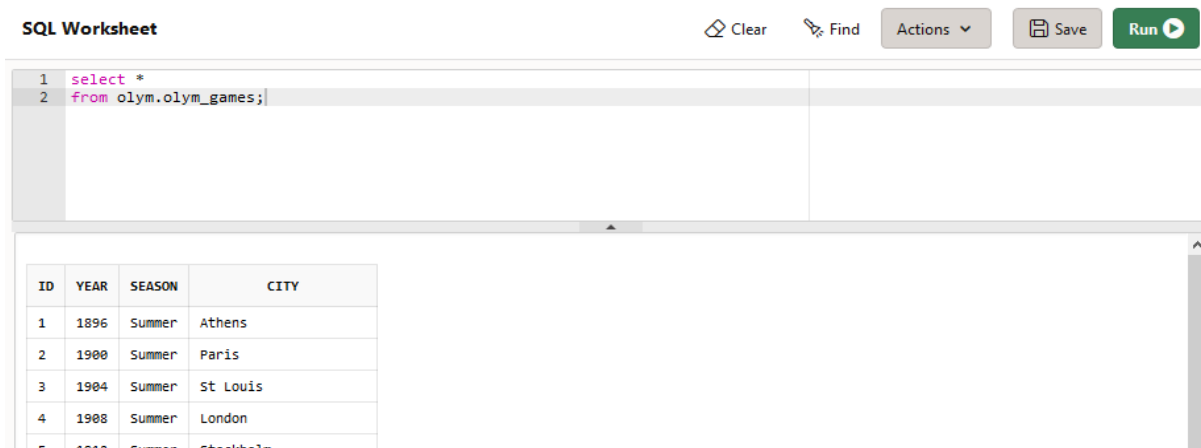
Once you have created an account and logged in you are ready to go. On the main (Home) page you will see that there are options to “Start Coding Now” (which is equivalent to selecting the “SQLWorksheet” option from the left-hand menu) or to “View Scripts and Tutorials”. There are many tutorials and a large number of scripts (identified by little green or red boxes above their names respectively) that it may be interesting to explore later.

However, for now it will be more useful (and you are less likely to get lost) to begin entering some simple SQL code to get a feel for the interface.

Chose the “Start Coding Now” button (or the “SQL Worksheet" menu option) and in the top window enter the following SQL command:

```
select *  
from OLYM.OLYM_GAMES;
```

Clicking “Run” provide you with a list of past summer Olympics up to 2008:



The screenshot shows the "SQL Worksheet" interface. At the top, there are buttons for "Clear", "Find", "Actions", "Save", and "Run". Below the buttons, the SQL query is entered in a text area:

```
1 select *  
2 from olym.olym_games;
```

Below the query area, the results of the query are displayed in a table:

ID	YEAR	SEASON	CITY
1	1896	Summer	Athens
2	1900	Summer	Paris
3	1904	Summer	St Louis
4	1908	Summer	London
5	1912	Summer	Stockholm

How do we know what tables (such as OLYM\_GAMES) already exist in this environment? Take a look at the “Schema” menu option to see these – there are around 10 schema that have been pre-populated by Oracle, including one related to the Olympic games (summer only, sadly for any winter sports enthusiasts among you). Each schema has a short-hand (alias) reference that we can use to access tables (or other objects) within that schema – so in the `from` clause above you see the format `<schemaAlias.tableName>`.

We can of course create our own tables.

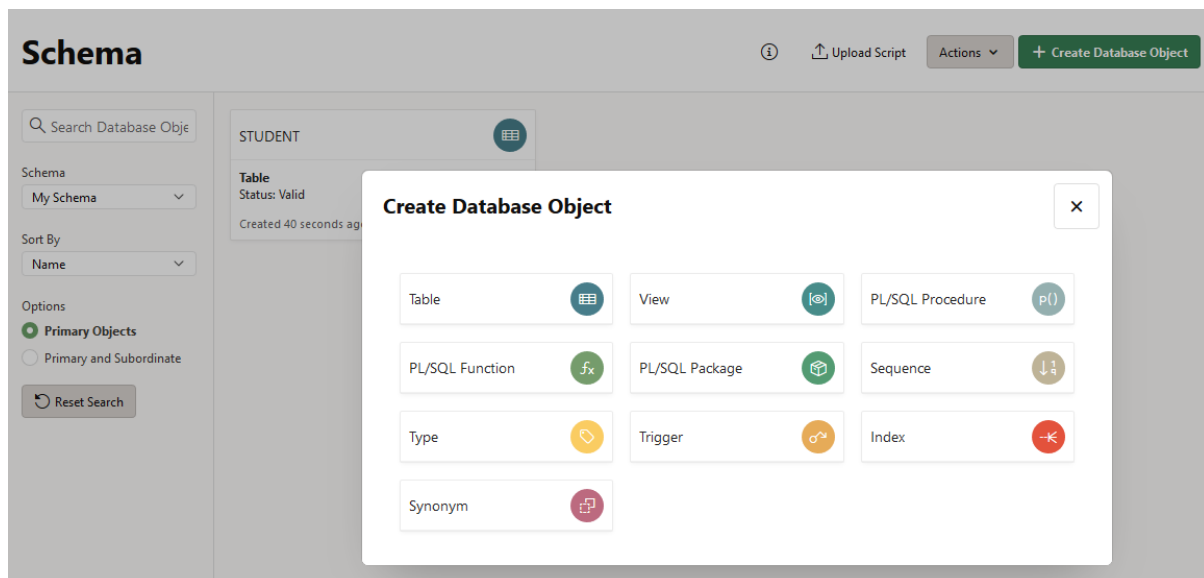
Let’s create a simple table to represent students, using the following statement:

```
CREATE TABLE Student (  
  StudentID VARCHAR2(14),  
  Surname VARCHAR2(40) NOT NULL,  
  StartYear NUMBER(4,0),  
  EndYear NUMBER(4,0),  
  Postcode VARCHAR2(9),  
  PRIMARY KEY (StudentID));
```

You will receive a “Table created” response if this executed correctly. (If you made any mistakes and need to run the command more than once, you will need to first use the “DROP TABLE” command to get rid of the old version of that table.)

Look again at the “Schema” option from the left-hand menu and then choose “MySchema” from the “Schema” drop-down menu. You should now see your new table (STUDENT) listed in this area (in fact, probably the only thing there!).

You should also see a “Create Database Object” button... Clicking this (‘wizard’) provides options to create all types of database objects – including PL/SQL functions and procedures, which may provide help with syntax next week...



Just as you will likely have stored a sequence of SQL commands in an external file and run these using the “@” in *SQLPlus*, so there is the option for running scripts (in addition to using interactive commands) within this *Live SQL* platform.

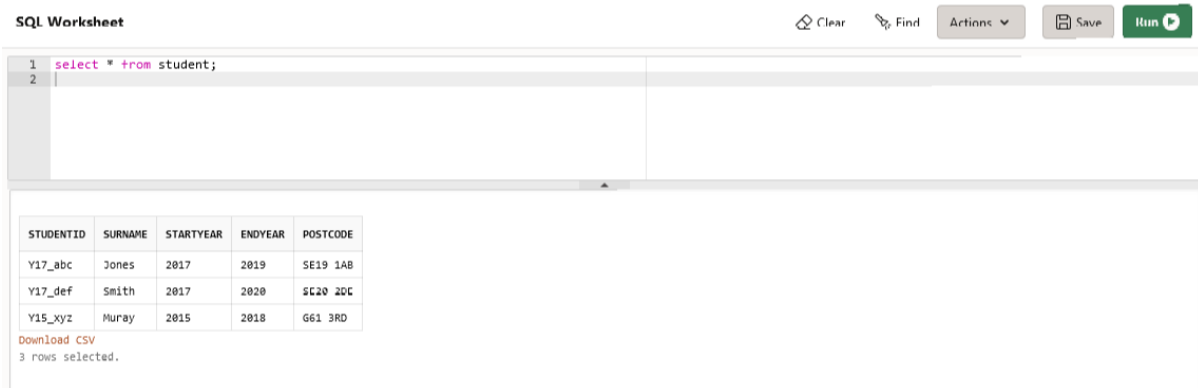
Create a text file named “newStudents.sql”, or similar, within which you can store commands to generate 3 records into your newly created table.

```
INSERT INTO Student VALUES ('Y17_abc', 'Jones', 2017, 2019, 'SE19 1AB');
INSERT INTO Student VALUES ('Y17_def', 'Smith', 2017, 2020, 'SE20 2DE');
INSERT INTO Student VALUES ('Y15_xyz', 'Muray', 2015, 2018, 'G61 3RD');
```

Choose the “My Scripts” menu option and then click on the “Upload Script” option to select and upload your file. The file does not have to have the suffix “.sql” but I find this helpful in terms of keeping track of file content type. Assuming you set the text up correctly, you will see that after the upload that three statements have been identified and when you hit “Run Script” you should get a message that three records (or however many rows you used) were created.

If you then return to the SQL Worksheet and run a SELECT command, you will find that these rows now exist in your table -

```
SELECT * FROM student;
```



Notice that here (both in the INSERT and SELECT statements) we did not need to specify a schema before the table name – this is because the table exists in *MySchema* (the default). However, for examples that have been pre-populated by Oracle you will need to remember the schema prefix. We may wish to use those examples/tables because they come with more ‘interesting’ data, but also because the tables and data that you create will NOT persist in the long term. (Oracle likely has 10,000s of users experimenting with this platform and as such will ‘purge’ your tables at regular intervals – I think after 24 hours, but you should check). All the more reason to store any SQL that you *do* create in the form of scripts which you can then quickly re-run to create the relevant tables, insert rows, etc. If you have created a set of commands in the interactive *SQL Worksheet* window you can use the “Save” option to store these as one of your scripts which you then have access to in the longer term. (Unlike tables and data, scripts persist between sessions and over time.)

In addition to the issue of table/data persistence, there are other limits that Oracle places on the usage of *Live SQL*. For example, you can’t have more than 2 active sessions open at any one time, nor can you retrieve more than 50,000,000 bytes of data (!) in a single session. Possibly of more relevance, you can only store up to 50 private scripts. You can see the ‘stats’ from your current session by looking at “My Session / Utilization”.

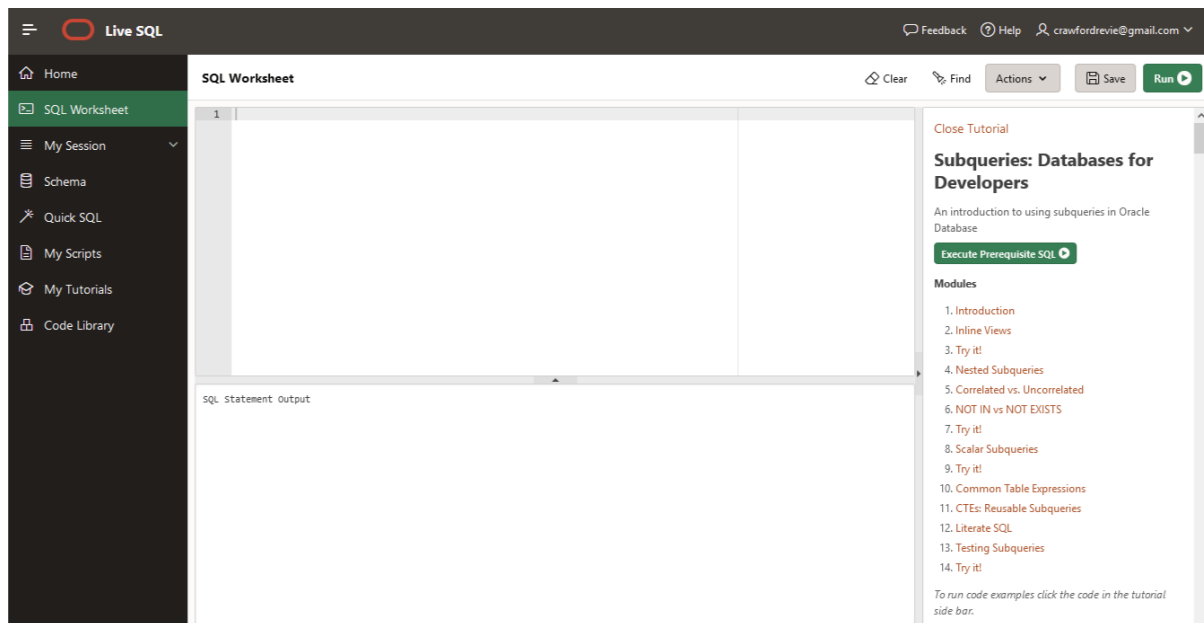
OK – so now that you are at least a bit familiar with the *Live SQL* environment, it may be a good idea to try out a few more ‘interesting’ queries.

## 2. Subqueries (Also an example on how to make use of the *Live SQL* tutorial feature)

I imagine that you have likely been introduced to these in John’s labs. However, both simple nested sub-queries and correlated sub-queries can involve a number of ‘subtleties’ that can make their use tricky. They can be very powerful so it is worth making sure that you feel comfortable using them. The exercise below also introduces an Oracle-created tutorial and it is useful to see how these are structured for future tutorial use.

Go back to the Home page in *Live SQL* and enter the word “subqueries” into the search box, then hit “View Scripts and Tutorials”.

In the Code Library you should see a TUTORIAL named “**Subqueries: Databases for Developers**” – you should click on this, and then choose the “Run Tutorial” button, at which point you will see a window similar to that shown below...



You will see an “Execute Prerequisite SQL” which basically contains the code that is needed to temporarily set up the environment (tables, indexes, views, etc.) for a given exercise. You need to run this script to execute those set-up commands.

**NOTE:** Typically, when you run these set-up scripts Oracle will drop all of the objects from your current schema and reset your session (e.g. previous commands). You should get a warning to that effect. So, for example, once you execute this the Student table that you just created and populated will be removed from *MySchema*. (This may also take up to 30s to run, so be patient!)

You should get a “Setup Complete” message that also tells you about database object dropped, etc. (Don’t worry, assuming that you have these saved as scripts it only takes a few clicks to restore them, if needed, in future.)

Once you have run the set-up you can follow this tutorial. Please don’t just use the “Insert into Editor” and hit Run(!) Instead, take a look at each query; try to **predict** what it is going to return and then make sure that you understand *how* the output was generated – particularly if the output is different to what you expected.

I have made a copy of the contents of the two tables (BRICKS and COLOURS) used in this tutorial at the top of the next page for quick reference.

Follow the 14 elements in this tutorial exercise, many of which you may be familiar with. The “NOT IN” versus “NOT EXISTS” (#6) make take a little while to get your head around, but is worth the effort. Also, the use of Common Table Expressions (CTE), that allow you to reference subqueries more coherently (#10 / #11), are worth understanding and knowing how to apply.

BRICK_ID	COLOUR
1	blue
2	blue
3	blue
4	green
5	green
6	red
7	red
8	red
9	-

**BRICKS**

COLOUR_NAME	MINIMUM_BRICKS_NEEDED
blue	2
green	3
red	2
orange	1
yellow	1
purple	1

**COLOURS**

There are four “Try it!” elements in the tutorial (at Steps #3, #7, #9 and #14). Actually [#14] is not very ‘interesting’ – instead you could use such an approach to find the number of different colours that are present in the BRICKS table – as you can see, the answer should be “3”.

Please store the SQL code that you create for these four little exercises in a single file named – “**CS992\_GrpX\_Lab1.sql**” (where the “X” should be the group to which you are allocated) – with annotation in the text so that I can see where each of the four elements begins/ends. You may even want to create a number of files with labels such as “...\_GrpX1...”, “...\_GrpX2...” for different group members. However, when it comes to group submissions for assessment I will only expect, and look at, one (agreed upon) file per group.

If any of your solutions does NOT give the same output as that shown in the tutorial, please still include it, with a note as to what you think might be going wrong.

### 3. Explore some other tutorials on aspects of SQL that interest you

If you complete the tutorial on subqueries fairly quickly, you may wish to use *Live SQL* to explore some other aspects of Oracle/SQL. Use the search options in the *Code Library* to check out some of those topics. Remember that the TUTORIAL entries will typically be a bit more self-explanatory/guided, while SCRIPTS come in many shapes and sizes (there are apparently over 1 million of them!).

Below I have provided a few suggestions, but don't feel limited to these if you want to explore something else that interests you.

**(NOTE:** In later labs we will be using *Live SQL* to work with PL/SQL and also exploring some of the analytic or *window* functions within SQL. Therefore, for the moment you might want to avoid tutorials that mention “PL/SQL” or have the word “Analytic” in their titles, as we will get to some of those in future sessions.)

#### **Joining Tables: Databases for Developers**

If you need a refresher on types of table **joins** (equi, left/right, cross, etc.) then this is a fairly simple and quite 'lite' tutorial. (If you found any aspects of the mini-lecture 1.4b a bit confusing and/or have not had much chance to work with 'joins' in the past, then this may be worth a look. The tutorial also highlights where the Oracle syntax differs from the ANSI specification, which may be useful for those who have used other RDBMS platforms in the past.)

#### **Aggregating Rows: Databases for Developers**

Again, not particularly challenging, but this tutorial provides a useful reminder as to how we can aggregate, group and summarise data in SQL.

#### **SQL/JSON Features**

This is an interesting tutorial introduction to some of the JSON-related extensions to SQL that Oracle has added to its DBMS over the past few years, in database versions 12.2. and 19c.

If you have not yet come across and/or used JSON formatted data this may be a bit tricky to get into, and this single tutorial probably needs the best part of the full two hours! However, it has lots of interesting material, including how to use GeoJSON to represent spatial information within databases, so it may be of interest for some of you to re-visit in the future.

(It also actually begins to introduce aspects of PL/SQL that we will come across next week, so don't worry too much about engaging in detail with this tutorial... It is included here more as a resource for those who already have some familiarity with using JSON data formats.)