TASKS

The tasks for this unit are to:

- Implement a sample of the tables identified from Normalisation
- Carry out exercises to alter and manipulate the database.
- To export SQL scripts for use on a new remote database.

For the purposes of these exercises you are to use Azure Data Studio and the docker instance that you created during setting up (see Setting Up Worksheet). You cannot carry out the implementation exercises unless you have completed the setting up workshop. Azure Data Studio will also be used to access the remote Microsoft SQL Server database issued to you for this module.

CONTEXT

When developing data driven applications you will need to be familiar with developing locally and deploying remotely. There may be times where you will not have a network connection and wish to work. Being able to minimise the impact of the loss of connection is therefore an important skill.

You can find information about how to use Azure Data Studio here
https://www.mssqltips.com/sqlservertip/6029/azure-data-studio-step-by-step-tutorial/

INSTRUCTIONS

Activity 1

Open your Docker desktop and start the container "mcr.microsoft.com/azure-sql-edge"

Open Azure Data Studio and re-connect to the database system as you did whilst setting up, but this time include the database you created last week: `COMP2001_Test`.

This means we do not need to include the database name in front of tables within the database.

Download the SQL file accompanying this workshop (DB *Workbook1.sql*). Using Azure Data Studio, and the connection to your local Microsoft SQL database, run the given SQL scripts.

Using the Servers left side bar in Azure Studio, expand the test database and the tables folder, view the tables, columns and constraints. Are they what you would expect?

Annotate the SQL file with comments so that you label important concepts that are demonstrated there. Use the checklist at the side to help you consider what is important to note.

- ✓ What does "Check" do?
- ✓ How would you use it?

HINTS:
The password was *C0mp2001!*
Ensure in your connection box the COMP2001_Test database you created is shown.

Activity 2

Taking the Student Module Enrolment System from the Normalisation and the Field Definition grid exercises, create appropriate tables in your test database using the following:

Grade table

```
CREATE TABLE dbo.Grade
(
    student_id INTEGER,
    module_code VARCHAR(3),
    mark TINYINT
        CHECK(mark BETWEEN 0 AND 100),
CONSTRAINT PK_Grade PRIMARY KEY (student_id, module_code),
CONSTRAINT FK_GradeStudent
     FOREIGN KEY (student_id)
     REFERENCES Student (student_id)
);
```

Why does this not run successfully and create a table 'Grade'?

Design and run the SQL to create the following 2 tables:

Student table

- student_id Integer
- firstname string (20 long) - mandatory
- lastname string (25 long) – mandatory
- Make student_id the Primary Key

Module table

- modulecode string (3 long)
- moduletitle string (25 long) – mandatory
- Make modulecode the Primary Key

Once you have the tables set up, copy the SQL statements below and run them in Azure Data Studio. What happens? Why?

```
INSERT INTO Student (student_id, firstname, lastname)
VALUES (103, 'Martin', 'Read');
INSERT INTO dbo.Student (student_id, firstname, lastname)
VALUES (108, 'Atkinson');
INSERT INTO dbo.Student (student_id, firstname, lastname)
VALUES (108, ,'Atkinson');
INSERT INTO dbo.Student (student_id, firstname, lastname)
VALUES ('S111', 'Andy', 'Smith');
```

Did some of the above statements run without errors?

Run the following to see what has been added to the Student table:

```
SELECT * FROM Student ORDER BY student_id;
```

Adjust the other SQL statements so that they also run successfully. Check that the data has been added to the table.

The structure of the Student table can be seen by running the following command:

```
EXEC sp_columns Student;
```

or

```
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME
= 'Student' ORDER BY ordinal_position;
```

Run the following code (correcting any errors).

```
INSERT INTO dbo.Module(moduecode, title)
VALUES ('DBS','Database Systems');
INSERT INTO dbo.Module(modulecode, title)
VALUES ('PR1','Programming 1');
INSERT INTO dbo.Module(modulecode, title)
VALUES ('PR1','Programming 2');
INSERT INTO dbo.Module(modulecode, title)
VALUES('IAI,'Intro to AI');
INSERT INTO dbo.Module(modulecode, title)
VALUES ('HCI','Human Computer Intaction');
```

Now create the Grade table above, modifying it to include a link to the module table.
Try to insert the following data (correct any errors).

```
INSERT INTO dbo.Grade (student_id, modulecode, mark)
VALUES (103, 'DBS', 72);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (103, 'IAI', 58);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (‘104’, 'IAI', 65);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (106, 'PR2', 43);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (107, 'PR1', 76);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (107, 'PR2', 135);
INSERT INTO dbo.Grade (student_id, module_code, mark)
VALUES (107, 'PR3', Null);
```

Create any test data needed in other tables.

Activity 3

ALTERING AND MANIPULATING YOUR DATABASE

Having created your database and carefully crafted the tables, you might review it and realise that you have missed something. Rather than delete the whole lot and start again you need to be able to alter and adapt.

Return to the original Order and OrderDetail tables from Activity 1. In the Order table, amend the column Customer so that it becomes CustomerID and is an integer.

Add a table called Customer that includes fields called CustomerID, Custname, Custaddress Custtelno. Make CustomerID a sequence (using IDENTITY) and the Primary Key.

Add some test data to Customer.

Add a foreign key constraint so that Order CustomerID is linked to the new Customer CustomerID Primary Key.

Add a new column to the OrderDetail table, the UnitPrice. This should be a suitable data type for a monetary value.

REMOTE DATABASE

For the purposes of this module only you have been allocated a database on the University instance of Microsoft SQL Server. The URL is DIST-5-505.uopnet.plymouth.ac.uk.

Your database log in details have been emailed to you separately. You MUST check them as soon as possible as errors in connecting cannot be rectified immediately prior to the deadline for submission of coursework.

NOTE: The database should only be used for COMP2001 learning purposes. This database is not provided to you for the rest of your studies and will only be active until the module has been completed.

Activity 4

Moving deployment to the remote server. In this exercise you will create a set of SQL scripts that will transfer your data structure to your remote database. Please note that only the structure of the tables can go across, the data does not.

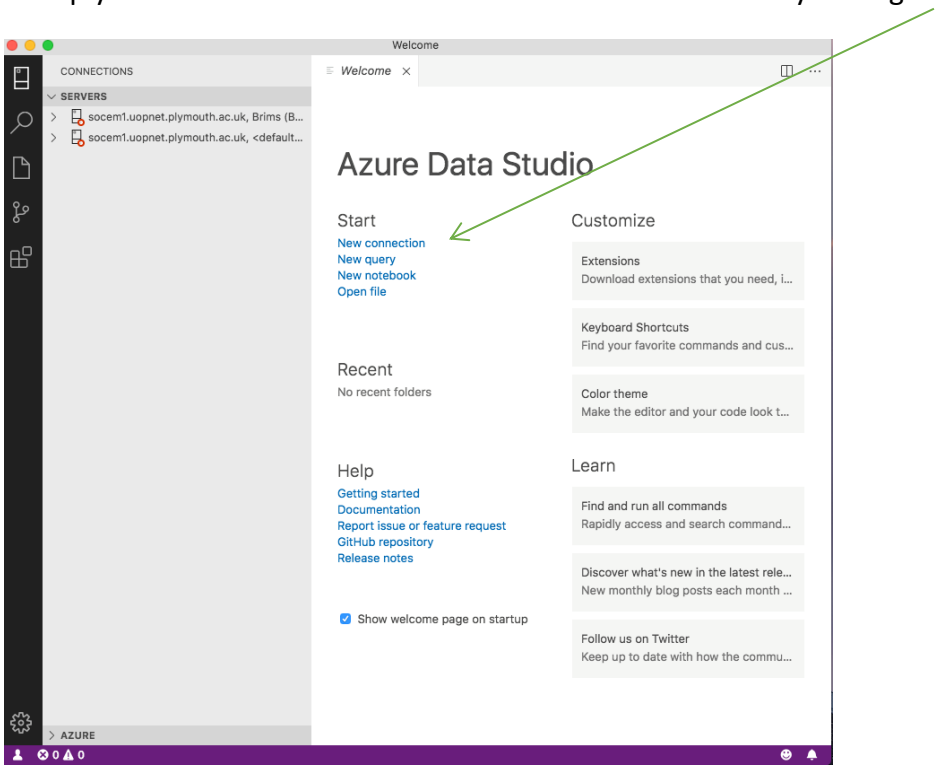
The remote database should be considered as your production database. In this exercise we will create tables but you must remove them after you have finished the exercise.

The first step is to export your chosen database. Right click the table you wish to export and select Script as Create.

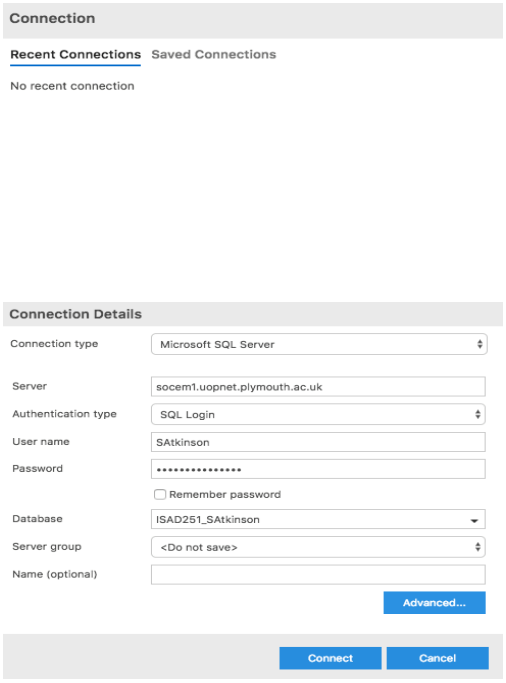
Do File -> Save As and save the resulting SQL file somewhere you can find it again.

Repeat the above for all your tables.

Set up your remote server connection in Azure Data Studio by adding a New Connection.



Complete the boxes in the following screen as right:



Server = DIST-6-505.uopnet.plymouth.ac.uk

Username = Your username – see email

Password = Your password – see email

Database = Enter your database name as per email – *do not use the drop down box as it will only show "master" and "tempdb" you do NOT want either of those.*

In the server pane, expand the Databases folder. You will have full access to the database labelled COMP2001_yourname. Please do not try and run any scripts elsewhere as you will not have permissions.

Create a new Query, ensure the connection is to your COMP2001 database and run a query you have saved. You should now see your database connected here.

Run the following code, as in the Setting Up practical:

```
CREATE database COMP2001_Test;
```

Why might it not run?

It is recommended that you practice your coding using the docker image and once you are happy with the results transfer the structure of your data tables to the remote server. Prior to the submission of your coursework you must clear down any tables not related to the coursework. You will lose marks if you do not keep your production database in a clean and tidy state.

If you hit problems with your Username/Password/Database access, please chat to Matt Seymour, the technician who looks after the labs. He can be found in SMB305.

Within a database we can create a schema to keep all related tables together in a logical way. A schema is just a container for a set of tables. In a New Query run the following:

CREATE SCHEMA test;

Now recreate the tables you created locally in your production database using the SQL you exported, but include the above schema name before the table name instead of dbo, e.g.

```
CREATE TABLE test.Grade, etc
```