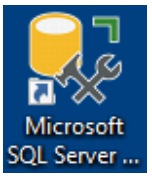
Laboratory Worksheet One

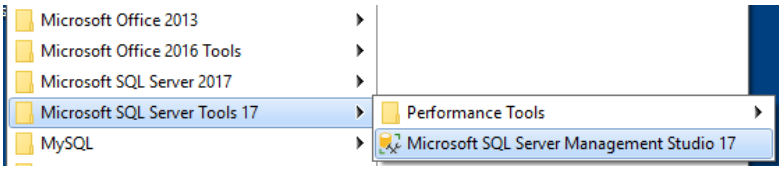
Microsoft SQL Server 2017 Express Edition is used for most of the practical work in the module. It is available on all Windows machines in the School Labs. However, you are **strongly advised** to install this software on your own machine so you can continue your learning and complete lab work outside scheduled lab times. You will find a document giving details of the download and install process (for **Windows machines**) in the Lab Resources folder on BlackBoard (*SQL Server 2017 - Download and Install Guidelines.pdf*).

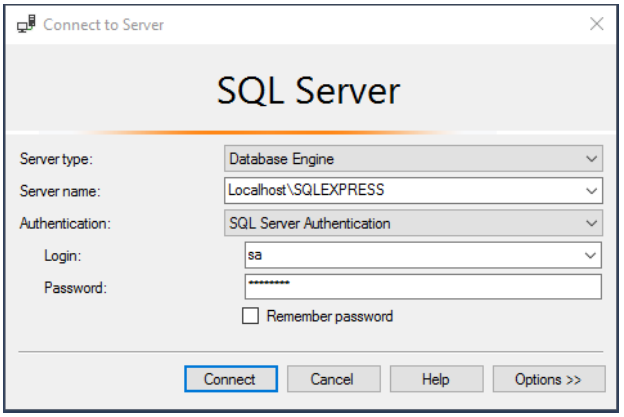


SQL Server Login (*on machines in School labs*)

Either: double click the **SQL Server Management Studio** icon on the desktop

Or: select **Microsoft SQL Server 2017** from All Programs and choose **SQL Server Management Studio**





To login, change (as necessary):

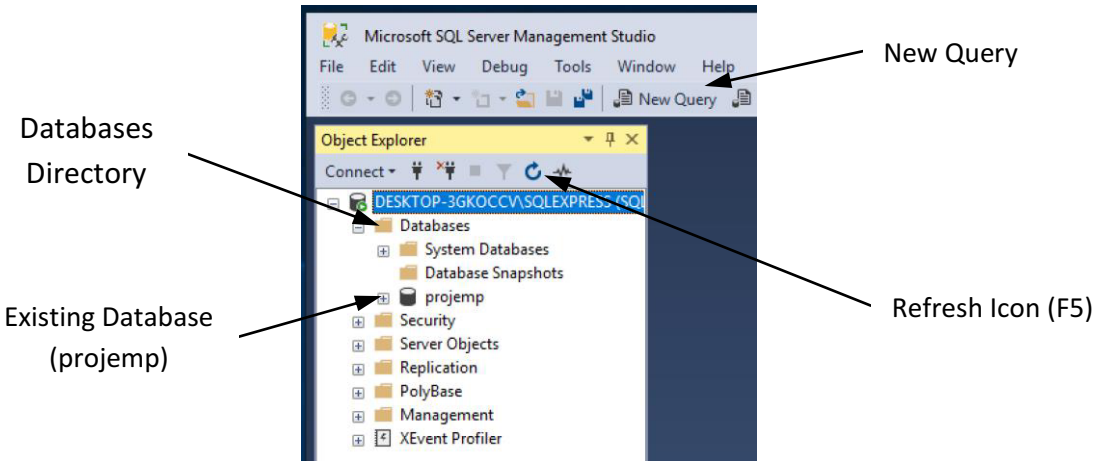
Server name to: localhost\SQLEXPRESS (should be the default)
Authentication to: SQL Server Authentication (as opposed to Windows Authentication)
Login: sa
Password: Labuser1 (case sensitive)

And click **Connect**

SQL Interface

Launch **SQL Server Management Studio** (SSMS) on your machine and login.

In **Object Explorer** expand the **Databases** directory (Click on **+** sign)



The example database used in the lab classes (projemp) should be listed in the database directory.

Create a New Database (called newdb2)

1. Right Click on the ***Database*** Directory in Object Explorer
2. Select ***New Database*** (at top of list)
3. Enter a name (**newdb2**) in the Database Name panel in the New Database window and press the **OK** button.
4. Note that **newdb2** is added to the list of Databases in the Object Explorer
   (It may be necessary to **'refresh'** before the new database can be seen - use **F5**, the **refresh icon** on the Object Explorer menu ribbon or **View | Refresh** from the main menu)

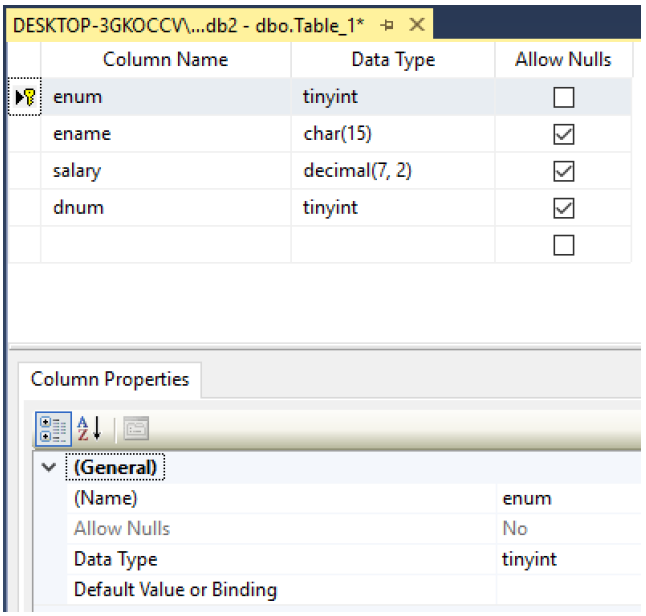
Create a new table in *newdb2* database

Tables can be created on screen or using SQL CREATE TABLE statement (covered in class)

To create a table on screen:

1. Expand ***newdb2*** directory (Click on + sign beside it)
2. Right Click on the ***tables*** directory
3. Select ***Table…*** (at top of list)

In table creation, create four attributes (columns) as shown



Column names are typed in

Data types are selected from the alphabetic drop-down list with **char** and **decimal** values changed in the **Length** (char) or **Precision & Scale** (decimal) column properties below:

Column Name	Data Type	
enum	tinyint	(Nulls not allowed)
ename	char(15)	change Length from default value 10
salary	decimal(7,2)	change Precision & Scale from 18 & 0
dnum	tinyint	

Note: Documentation on T-SQL, including information on datatypes is available by following the link provided in the Lab Resources folder in BlackBoard or directly at https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-2017

1. Set **enum** as the **primary key** (must contain unique values AND not accept NULL values) by Right-clicking on panel to left of **enum** and select top option **Set Primary Key**

2. The table is saved either by closing the table window and clicking **Yes** when asked to save
   OR clicking the File menu then the **Save ….. dbo.Table_1** option
   OR using **Ctrl + S**
3. This prompts a Choose Name window – Enter **emp** and press **OK**
4. Expand the **newdb2** tables directory to see that **emp** is now a table listed

The design can be viewed/edited by right clicking on the table and selecting **Design**
(If Editing changes are not allowed then table must be deleted with *Right Click* and *Delete* then recreated)

Create a *dept* table in *newdb2* database

Using the same approach as above, create the following table:

Column Name	Data Type	
dnum	tinyint	(Nulls not allowed)
dname	char(15)	
location	char(15)	

Make **dnum** the **primary key** and save table as **dept**

It may be necessary to **'refresh'** before the new table is shown in Object Explorer.

Note that **Refresh** only refreshes those levels of the hierarchy in Object Explorer ***below*** the level highlighted. For example, to see the table just created, ensure that **newdb2**, or a level above it, is highlighted before refreshing.

Putting Referential Integrity between the tables (via dnum)

Copy and paste the following code to make **dnum** a Foreign Key in the **emp** table:

```
alter table emp
add constraint fk_emp foreign key(dnum) references dept(dnum);
```

When the statement is executed **referential integrity** is established between **dnum** *(Foreign Key in emp)* and **dnum** *(Primary Key in dept)*.

In other words, a **relationship** has been established between the two tables - through the Foreign Key. For any employee in the **emp** table, **dnum** gives their department number, **referencing** the corresponding **dnum** entry in the **dept** table, which holds information on that department.

Adding Records into the dept table

- With referential integrity enforced records **CANNOT** be added to the **emp** table until **dept** records are added.

  In particular, a value for **dnum** cannot be entered in the **emp** table (Foreign Key) **unless** a corresponding record for **dnum** already exists in the **dept** table.

- Records can be entered using an SQL INSERT query or on screen
  (we are using a quick method for now – in future we will use an SQL query statement)

1. To enter records onscreen, right click on the table name **dept** and select **Edit top 200 rows.** This gives a table (with **NULL** values for first row).
2. Add the following values by overwriting the **NULLs** in each successive row:

dnum	dname	location
11	sales	Belfast
12	admin	Bangor

3. Close the window and view the table using Right Click on the **dept** table and selecting **Select Top 1000 rows**

Adding Records into the *emp* table

1. Repeat the record entry for emp as you did with the dept table above:

enum	ename	salary	dnum
1	smith	20000	11
2	brown	25000	12
3	white	30000	11
4	jones	28000	11
5	kelly	40000	12
6	smith	18000	12
7	green	21000	11
8	doyle	28000	12
9	brown	38000	11
10	magee	27000	11

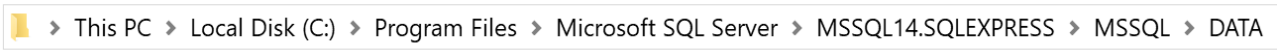
2. Close the window and view the table.
3. Primary key values must be unique. If you try to add a primary key value that is already entered, the record cannot be saved, and a primary key violation message appears. Try adding a repeated primary key value to check this
4. Close the query window
5. View the table by right clicking on the emp table and selecting Select Top 1000 rows (Again this is a shortcut – in future we will be using SQL statements to view data)

Detaching and Attaching a Database

One of the ways of keeping a database that you have created ***(if you are using a machine in the School Labs, files WILL BE DELETED on logoff)*** is to **detach** it and copy the database file (.mdf)

All .mdf files reside in

C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA



.mdf files CANNOT be removed until the active connection is closed:

1. Right Click on **newdb2**, select **Tasks** then **Detach**
2. Click the **Drop** tickbox for **newdb2** then press **OK**

   (Note that newdb2 has disappeared from the Databases list in Object Explorer)
3. Go to C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA
4. **Copy** **newdb2.mdf** and **Paste** onto the desktop. To save the **.mdf** file to a memory stick or to your P: drive, **Copy** the file again, from the desktop, and **Paste** into your destination folder. This process will ensure that your database file will have the correct permissions associated with it at each stage.

In general, always use **'Copy & Paste'** when moving database files - do **NOT** use **'Drag & Drop'.**

If necessary, click 'continue' if you are asked for administrator permissions.

Note that there is also a log file **newdb2_log.ldf** containing information on the database activity – in a real database this would have to be copied as well, but for a static database used in the labs this is not necessary

5. Now delete **newdb2.mdf** and **newdb2_log.ldf** from the DATA directory (**after copying**)

Attaching an mdf file

To **attach** a database using an mdf file (or mdf plus ldf file) reverse the detach process above.

1. **Copy** the .mdf file (and .ldf if available) from your folder and **Paste** directly into the DATA directory on the C: drive (it is not necessary to use the desktop in the detach process).
2. Right click on **Databases** then **Attach**
3. Click the **Add** button and select **newdb2.mdf** from the list
   If the .ldf file is *Not Found* then highlight it and click the **Remove** button in the lower panel
   (The database will recreate this file – in a real database which is dynamic the .ldf file is very important for database recovery on failure)
4. Click the **OK** button and check that **newdb2** is now on the Databases list

Using SQL to Create a new Database and Tables

We will create a new database (called **ed2**) with a structure similar to **newdb2** previously created on screen, but this time using SQL commands.

First, some basic information on SQL

T-SQL (Transact SQL) is the SQL version used in SQL Server and has a range of commands to:

- create a new database
- create a new table (structure)
- alter the table structure
- add new records
- update (change) an existing record
- delete a record
- retrieve existing records

The general syntax for table creation is:

```
CREATE table tablename
( attribute1   datatype (NULL/NOTNULL),
  attribute2  datatype (NULL/NOTNULL),
   ……..
  attributen  datatype (NULL/NOTNULL),
  CONSTRAINT name  PRIMARY KEY (PK_attribute)
);
```

1. If it is not already open, launch Microsoft SQL Server 2017 and login following the procedure outlined at the start of this worksheet.

2 Click on the **New Query tab** in the top menu (or *Ctrl – N*) to create a query window (remember to close it after use, saving to your P: drive as an SQL file if required using *File | Save SQLQueryX As*)

3. Enter **create database ed2;** and press Execute Tab (or *Right Click* and *Execute* or *F5*)

4. Make **ed2** the current database by executing **use ed2;** in a new query window or overwrite previous query
   (Notice that **ed2** becomes the current database in the Database List in the top menu)

5. In a new query window enter the following create table statement (copy and paste)

```
create table emp
( enum tinyint not null,
  ename char(15),
  salary decimal(8,2),
  dnum tinyint,
  constraint PK_emp   primary key (enum)
);
```

   This creates the table structure for the **emp** table – check the table has been created – this may require a right click refresh on the database

6. Now create a department table called **dept** using a **CREATE TABLE** statement with appropriate data types and sizes and save the working SQL file (**File | Save SQLQueryX.sql As …**) to your P: drive.
   (You can copy, paste and adapt the above create table statement for **emp**)

Attribute	Type of data	Description
dnum	whole numbers between 10 to 99	Department Number
dname	words of up to 12 letters	Department Name
city	words of up to 15 letters	City
budget	money up to 999999.99	Budget
mnum	whole numbers from 1 – 99	Manager Number

make **dnum** the primary key

Add Referential Integrity between the Tables

1. Copy and paste the following code to put referential integrity between **dnum** *(Foreign Key in emp)* and **dnum** *(Primary Key in dept)*.

```
alter table emp
add constraint fk_emp foreign key(dnum) references dept(dnum);
```

NOTE: referential integrity is always added to the Foreign Key (FK) attribute to reference a Primary Key (PK) in another table. This will not work if:

- PK and FK do not have the same data type and size
- any **emp** records exist with FK dnum values not found in PK dnum of **dept** table.

Create a Manager Table and add Referential Integrity

1. Create a manager table (copy, paste and change the **CREATE TABLE** statement for **emp**)

```
manager (mnum, mname, office, phonenumber)
```

Where **mnum** is defined exactly as **mnum** in dept table
mname allows for surnames
office - an example value would be 16C26 (**char** can take alphanumeric characters)
phonenumber: define as 11 characters (i.e. not as an integer)

Primary key is **mnum**

2 Use **ALTER TABLE** to place referential integrity between **mnum** of **dept** (FK) and **mnum** of **manager** (PK)
(You can copy, paste and change the **emp** dept referential integrity **ALTER TABLE** statement above).

For Information Only – Some Useful ALTER TABLE Commands

The **ALTER TABLE** command is useful if you need to change the table design:
Some sample statements to be viewed but **not** implemented

Add the primary key dnum to dept (if not put in the create table statement):
alter table dept
add constraint pk_dept primary key (dnum);

Change a datatype on an attribute (e.g. make dname 20 characters):
alter table dept
alter column dname char(20);

Add a new column location of 15 characters long;
alter table dept
add location char(15);

Delete location attribute:
alter table dept
drop column location;

Make dnum not null:
alter table dept
alter column dnum tinyint not null;

Dropping a PK called PK_dept:
alter table dept
drop constraint pk_dept;

Adding Records to a Table

Since referential integrity has been created between **emp.dnum** (FK) and **dept.dnum** (PK) you CANNOT add any employee records until the corresponding department record has been added. (You can try this out to check)

SQL **INSERT** command syntax

INSERT INTO tablename (attribute list with commas in between)
VALUES (value list with commas in between);

NOTE: Because referential integrity is now in place you cannot add employees until departments are added AND you cannot add departments until managers are added.
So, add MANAGERS first, then DEPARTMENTS and finally EMPLOYEES

1. Add a manager execute the following statement

INSERT INTO manager (mnum, mname, office, phonenumber)
VALUES (1, 'vincent', '10G15', '02812345678');

NOTE that:

- naming the attributes in a list is not necessary where a whole record is being added
- character values MUST have inverted commas round them (single or double)
- numeric values do NOT have inverted commas round them

So:

```
INSERT INTO manager
VALUES (1, 'vincent', '10G15', '02812345678');
```

would also work in this case (no attribute list required for whole record)

2. Now edit the SQL statement to add the following records (i.e. just change the **VALUES** statement and rerun)

   2, mccormick, 17B01, 02823456789 (remember the inverted commas!)
   3, thompson, 5H23, 02834567890

3. Now add the following records to **dept** using individual **SQL INSERT** statements:

   10, sales, belfast, 58750.00, 1
   11, admin, coleraine, 39500.00, 2
   12, finance, belfast, 64700.00, 3

4. Now add the following records to **emp** using individual **SQL INSERT** statements:
   (if you have not got time use the *Edit Top 200 Records* shortcut)

   1, smith, 25000.00, 10
   2, jones, 27000.00, 12
   3, brown, 31000.00, 10
   4, white, 20000.00, 11
   5, kelly, 28000.00, 12
   6, green, 41000.00, 10
   7, black, 30000.00, 11

Examining the Tables

Retrieval of whole tables can be achieved simply using ***** (meaning all attributes) as in

```
SELECT *        (Statement could be on 1 line – only on 2 lines for clarity)
FROM emp;
```

Look at all the tables using this command (changing table names)

Deleting and Updating Records

General Syntax of **DELETE**:

```
DELETE FROM tablename
WHERE attribute compared to value;        (compared to can be =, >, <, >=, <= or < >)
```

General Syntax of **UPDATE**:

```
UPDATE tablename
SET attribute = new value
WHERE attribute compared to value;
```

Modifying the Information in the Database

Your database (**ed2**) should have the following table structure (PK underlined)

emp(<u>enum</u>, ename, salary, dnum)
dept(<u>dnum</u>, dname, city, budget, mnum)
manager(<u>mnum</u>, mname, office, phonenumber)

1. Change the salary of employee number 4 to 23000:

```
update emp
set salary = 23000
where enum = 4;
```

2. Change the office of the manager named 'thompson' to '5J01' (use **UPDATE**, check the updated value and save the SQL statement as an sql file).
3. Delete the employee named 'green' (use **DELETE**, check the updated value and save the SQL statement as an sql file).

Finally – Detach the database and Store the .mdf file

All .mdf files reside in

C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA

Remember, .mdf files CANNOT be removed until the active connection is dropped:

1. Right Click on **ed2**, select **Tasks** then **Detach**
2. Click the **drop** tickbox for **ed2** then press **OK**
   (Note that **ed2** has disappeared from the Databases list in Object Explorer)
3. Go to

   C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA
4. **Copy** the **ed2.mdf** file, via the desktop, to your memory stick or P: drive.
   (Note that there is also a log file **ed2_log.ldf** containing information on the database activity – in a real database this would have to be copied as well but for a static database used in the labs this is not necessary).
5. Now delete **ed2.mdf** and **ed2_log.ldf** from the DATA directory (**after copying**).

To use the **ed2** database again it must first be **attached**.

- Place the database files (**ed2.mdf** and **ed2_log.ldf** (if available)) in the above DATA directory,
- Launch SQL Server 2017 and right click on **Databases** in Object Explorer
- Click **Attach**, click **Add**, find **ed2.mdf** in the list and click **OK**
- **ed2** should now show on the list of available **Databases**.

NOTE: use **'Copy & Paste'** when moving database files - do **NOT** use **'Drag & Drop'.**