INFO90002 Week 3 Lab

Objectives:

- Review the Case Study & your conceptual model
- Use MySQL modelling tool to build a logical model
- Introduction to SQL

Section 1. Data Modelling

1.1. Review the case study and your pen and paper conceptual data model from Week 2.

Case Study: The Department Store

This database is the central component of an information system used to manage a department store that specialises in camping and hiking equipment. The store has several departments. For each department we must record its name and unique department id, phone number, and which floor it is on. Each department has several employees working for it. Each department has a manager. A manager can manage one or more departments.

About each employee we record their first name, last name, their date of birth, a unique employee id, their annual salary, which department they work for and which other employee is their boss. The General Manager of the Department Store has no boss.

The items that the store sells each have a name and id, a type, a colour and the retail price. Whenever a department sells items to customers we record the date of sale, which item was sold, the quantity of each were sold and which department sold it. Each sale may contain one or more items. Each sale is made by one department within the store.

Items are delivered to the store by suppliers. Each delivery from a supplier contains one or more items delivered to one or more departments within the store. We record which supplier made the delivery and to which department, and the wholesale price of each item. For each supplier we record a unique supplier id, name and contact phone number.

1.2 Use MySQL Workbench modelling tool to construct a logical data model identifying primary and foreign keys and the required attributes for each entity.

1 MySQL Workbench Data Modelling

MySQL Workbench has three sections: the default window, the modelling window and the migration window.

TASK 1: Launch MySQL Workbench

When you launch MySQL Workbench you will be in the default window. This is highlighted in the right frame of MySQL Worksheet and circled in red below:

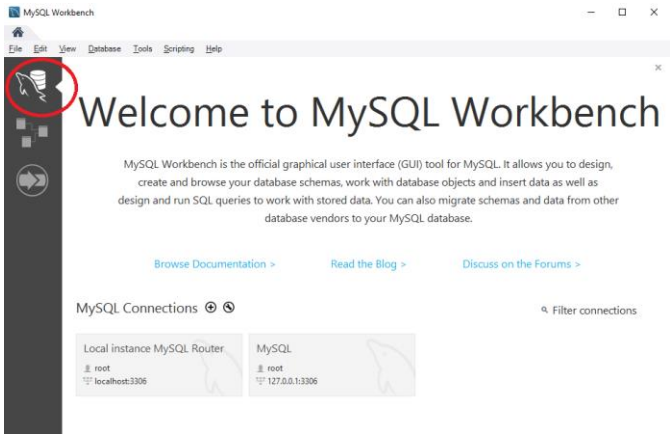

Figure 1. The default MySQL Workbench window

You will need to select the data modelling window

Figure 2: The Modeling icon

This will now be the highlighted window with the pointer:

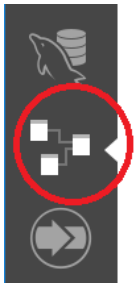

Figure 3: The active window for data modelling

And your Window will have changed to a different view:

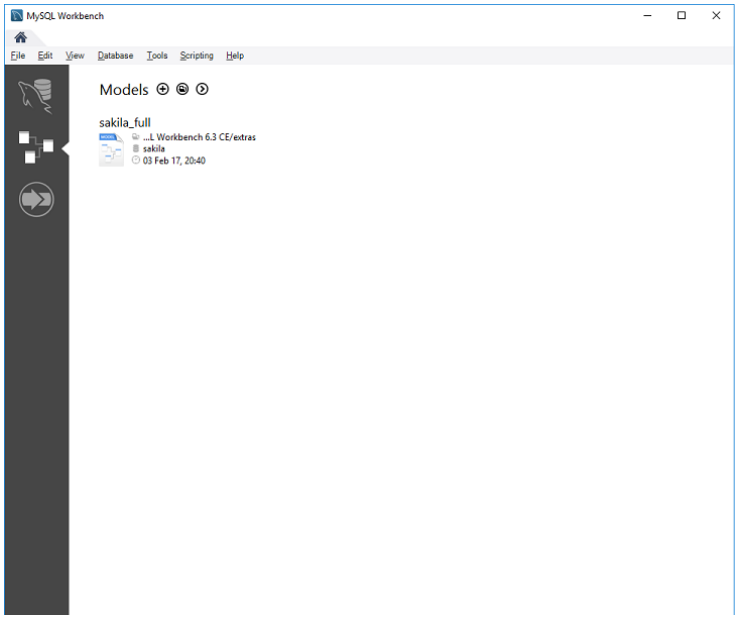

Figure 4: The full screen of the data modelling module

TASK2: Click the Add model (+) symbol next to the Model at the top left of the MySQL Window.

This will launch a new modelling window:

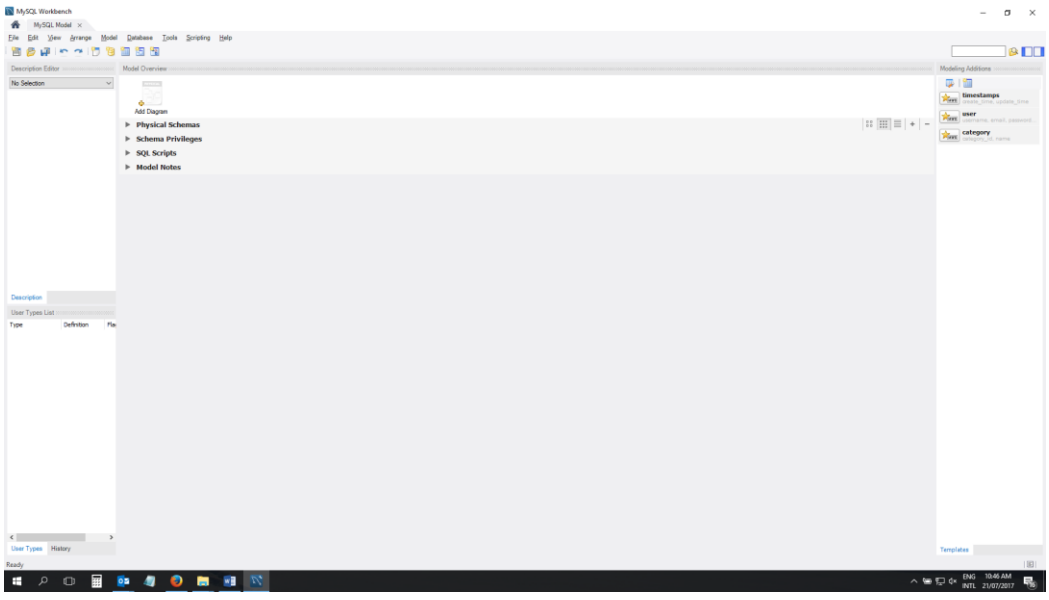

Figure 5: The new modelling window

TASK 3: Click the “Add Diagram” icon to add a new diagram:

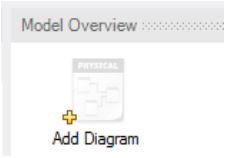

Figure 6: The add diagram icon

This will bring a second tab and a diagram window:

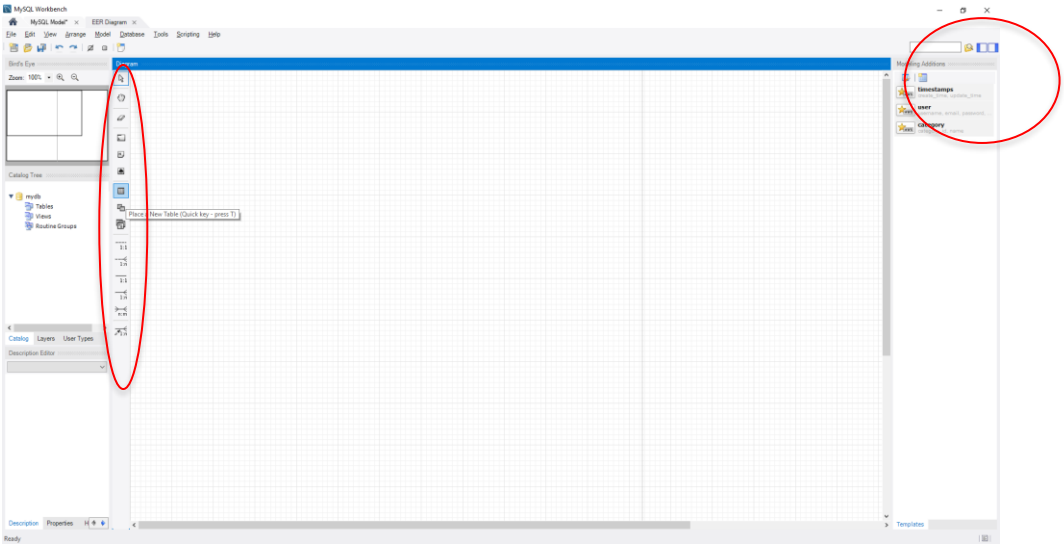

Figure 7: The diagram with modelling tools on the left hand side of the canvas. Note the minimizing of windows icon in the top right hand corner of the canvas

TASK 4: Make more space for your diagram. Remove the left and right side columns of your modelling canvas:

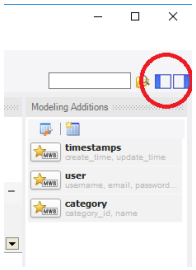

Figure 9: Highlight of the icon to minimize the right and left columns of the modelling canvas

Adding tables to your model

TASK 5: Using your mouse click once on the table icon and drag it to your diagram

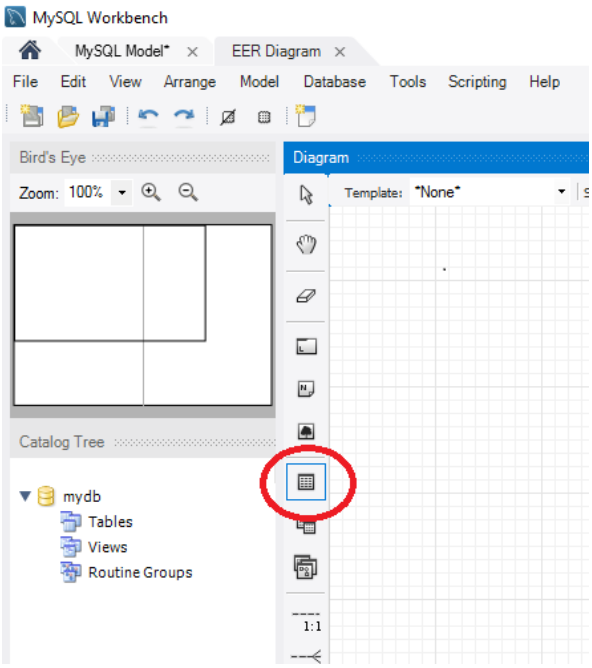

Figure 10 The add table tool

TASK 6: Drop the table on your diagram

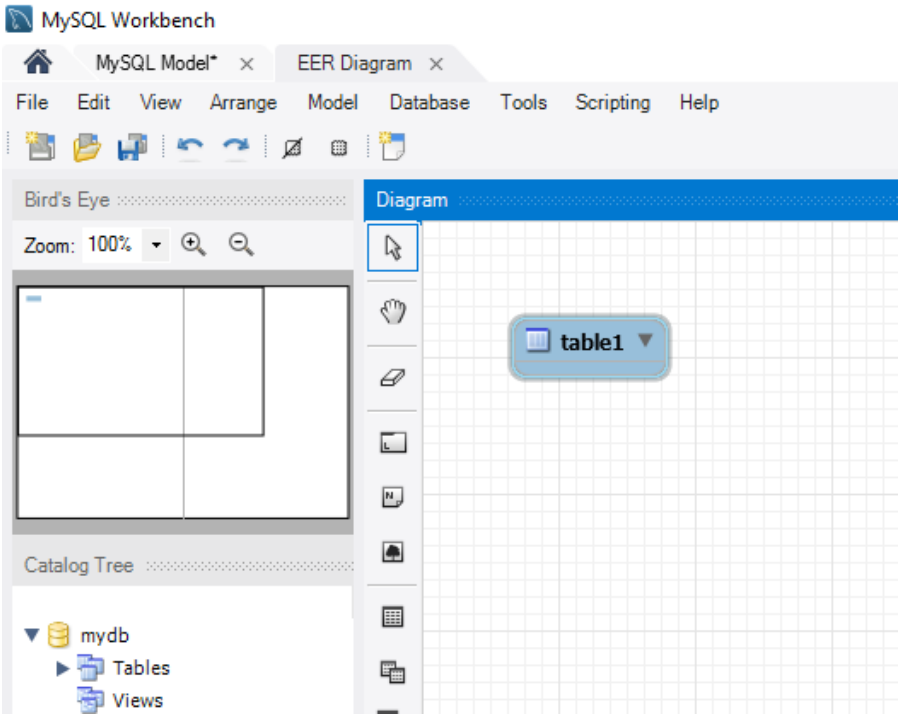

Figure 11 The table has now been placed on the diagram

Task 7: Double click the table and the table wizard appears at the bottom of the application

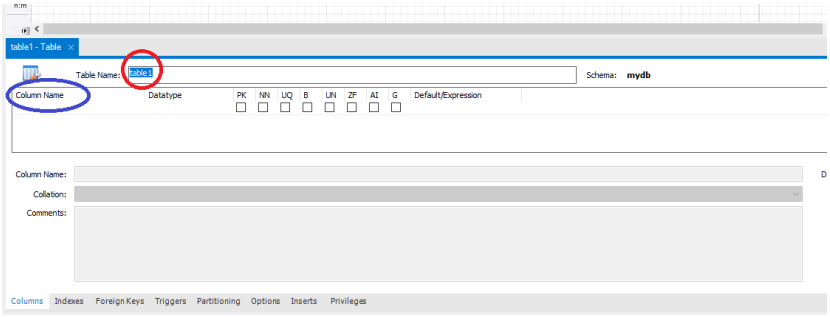

Figure 12 The Table Wizard dialog box

The Table name is highlighted

Section 2 SQL

Using MySQL Workbench, connect to your database on the MySQL Server

2.1 Show the metadata about the Department table

```
DESC Department;
```

Hint: if you get an error message or can't find the Department table, check that you ran the Setup Script in week 2 you may also need to tell the database to use the correct schema

```
use <username>;
```

Structure of SQL statements

The structure of a SELECT statement is below:

```
SELECT (select list)
FROM (from list)
WHERE (filtering list) -- optional
GROUP BY (grouping list) -- optional
HAVING (group qualifications) -- optional
```

To select all columns from a table we use the SQL shorthand "*"

To select from the Department table, enter the following SQL:

```
SELECT *
FROM department;
```

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
2	Books	1	81	4
3	Clothes	2	24	4
4	Equipment	3	57	3
5	Furniture	4	14	3
6	Navigation	1	41	3
7	Recreation	2	29	4
8	Accounting	5	35	5
9	Purchasing	5	36	7
10	Personnel	5	37	9
11	Marketing	5	38	2
NULL	NULL	NULL	NULL	NULL

2.2 Type the SQL query to select all rows and columns from the employee table.

You should see a result like this:

EmployeeID	FirstName	LastName	Salary	DepartmentID	BossID	DateOfBirth
1	Alice	Munro	125000.00	1	NULL	1966-12-14
2	Ned	Kelly	85000.00	11	1	1970-07-16
3	Andrew	Jackson	55000.00	11	2	1958-04-01
4	Clare	Underwood	52000.00	11	2	1982-09-22
5	Todd	Beamer	68000.00	8	1	1965-05-24
6	Nancy	Cartwright	52000.00	8	5	1993-04-11
7	Brier	Patch	73000.00	9	1	1981-10-16
8	Sarah	Fergusson	86000.00	9	7	1978-11-15
9	Sophie	Monk	75000.00	10	1	1986-12-15
10	Sanjay	Patel	45000.00	6	3	1984-01-28
11	Rita	Skeeter	45000.00	2	4	1988-02-22
12	Gigi	Montez	46000.00	3	4	1992-03-20
13	Maggie	Smith	46000.00	3	4	1991-04-29
14	Paul	Innit	41000.00	4	3	1998-06-02
15	James	Mason	45000.00	4	3	1995-07-30
16	Pat	Clarkson	45000.00	5	3	1997-08-28
17	Mark	Zhang	45000.00	7	3	1996-10-01

Filtering Results

To select only some columns, we specify the columns we want in the query. Separate each column name with a ",". If you describe the Department table we can see attributes Name and Floor, amongst others.

To select only these two columns in the Department table, enter this SQL:

```
SELECT Name, Floor
FROM department;
```

You should see a result set like this:

	Name	Floor
	Management	5
	Books	1
	Clothes	2
	Equipment	3
	Furniture	4
	Navigation	1
	Recreation	2
	Accounting	5
	Purchasing	5
	Personnel	5
	Marketing	5

This is known as a *projection* to filter the result set by columns.

2.3 Type the SQL query to select the first name, last name and departmentid in the Employee table.

	firstname	lastname	departmentid
	Alice	Munro	1
	Ned	Kelly	11
	Andrew	Jackson	11
	Clare	Underwood	11
	Todd	Beamer	8
	Nancy	Cartwright	8
	Brier	Patch	9
	Sarah	Fergusson	9
	Sophie	Monk	10
	Sanjay	Patel	6
	Rita	Skeeter	2
	Gigi	Montez	3
	Maggie	Smith	3
	Paul	Innit	4
	James	Mason	4
	Pat	Clarkson	5
	Mark	Zhang	7

Until now we have selected **all** the rows in a table. Most times we don't want to retrieve all rows.

In SQL we do this by using a *selection condition* on our query. This is also known as a *selection*. If for example, we wished to list all Departments that are listed on the second floor:

```
SELECT *
FROM department
WHERE floor = 2;
```

	DepartmentID	Name	Floor	Phone	ManagerID
	3	Clothes	2	24	4
	7	Recreation	2	29	4
	NULL	NULL	NULL	NULL	NULL

The SQL keyword 'WHERE' lists any horizontal filter we wish to put on the query.

How do we find out all department names that start with M? To do this we use the wildcard '%'. % stands for any character or characters. When you use % you need the SQL word LIKE. To display all departments that start with M we type:

```
SELECT *
FROM department
WHERE Name LIKE 'M%';
```

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
11	Marketing	5	38	2
NULL	NULL	NULL	NULL	NULL

2.4 Type the SQL query to return the first and last names and department id of all employees who earn less than $55000.

```
SELECT firstname, lastname, departmentid
FROM employee
WHERE Salary < 55000;
```

Your result set should look like this

firstname	lastname	salary
Paul	Innit	41000.00

Multiple Conditions

Sometimes we may need to filter the result set by having more than one condition met. For example, if we wish to list all the Departments that start with M and whose manager ID is 1

```
SELECT *
FROM department
WHERE Name like 'M%'
AND ManagerID = 1;
```

Both conditions must be true, and in this case only 1 row is returned:

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
NULL	NULL	NULL	NULL	NULL

However, if we change the AND to OR the result set changes. Two rows are returned.

When we use an OR condition, only one condition need be true for a row to be returned.

```
SELECT *
FROM department
WHERE Name like 'M%'
OR ManagerID = 1;
```

The query lists all departments starting with M, as well as all departments where the ManagerID is equal to 1.

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
11	Marketing	5	38	2
NULL	NULL	NULL	NULL	NULL

2.5 Type the SQL query that lists all employees who work in DepartmentID 11 AND who earn greater than 55000.

firstname	lastname	departmentid	salary
Ned	Kelly	11	85000.00

Then change the AND to OR and note the difference in the result set. Notice Clare Underwood's department and salary

firstname	lastname	departmentid	salary
Alice	Munro	1	125000.00
Ned	Kelly	11	85000.00
Andrew	Jackson	11	55000.00
Clare	Underwood	11	52000.00
Todd	Beamer	8	68000.00
Brier	Patch	9	73000.00
Sarah	Fergusson	9	86000.00
Sophie	Monk	10	75000.00

2.6 In MySQL Workbench show the metadata about the Department and Employee tables

```
DESC Department;
DESC Employee;
```

Let's do some more mathematical operators and functions

To select all departments that are above the first floor we would type

```
SELECT *
FROM department
WHERE Floor > 1;
```

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
3	Clothes	2	24	4
4	Equipment	3	57	3
5	Furniture	4	14	3
7	Recreation	2	29	4
8	Accounting	5	35	5
9	Purchasing	5	36	7
10	Personnel	5	37	9
11	Marketing	5	38	2
NULL	NULL	NULL	NULL	NULL

Or find out which departments are NOT on the fifth floor

```
SELECT Name, Floor
FROM department
WHERE Floor != 5;
```

OR

```
SELECT Name, Floor
FROM department
WHERE Floor <> 5;
```

Note that != and <> both mean 'Not Equal To'

Name	Floor
Books	1
Clothes	2
Equipment	3
Furniture	4
Navigation	1
Recreation	2

Order By

We can order the result set by any column (it does not have to be in the SELECT clause). The ORDER BY forces the result set to be ordered by the values of one or more columns.

```
SELECT Name, Floor
FROM department
WHERE Floor != 5
ORDER BY Floor;
```

Name	Floor
Books	1
Navigation	1
Clothes	2
Recreation	2
Equipment	3
Furniture	4

The default sort order is from the smallest value to largest (1-10 or A-Z). You can explicitly state this by typing ASC (short for Ascending order). To sort from largest value to smallest you would enter DESC (short for Descending order).

```
SELECT *
FROM department
ORDER BY Floor DESC;
```

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
8	Accounting	5	35	5
9	Purchasing	5	36	7
10	Personnel	5	37	9
11	Marketing	5	38	2
5	Furniture	4	14	3
4	Equipment	3	57	3
3	Clothes	2	24	4
7	Recreation	2	29	4
2	Books	1	81	4
6	Navigation	1	41	3
NULL	NULL	NULL	NULL	NULL

You can order by more than one column and in different order for each column:

```
SELECT *
FROM department
ORDER BY Floor DESC, DepartmentID ASC;
```

DepartmentID	Name	Floor	Phone	ManagerID
1	Management	5	34	1
8	Accounting	5	35	5
9	Purchasing	5	36	7
10	Personnel	5	37	9
11	Marketing	5	38	2
5	Furniture	4	14	3
4	Equipment	3	57	3
3	Clothes	2	24	4
7	Recreation	2	29	4
2	Books	1	81	4
6	Navigation	1	41	3
NULL	NULL	NULL	NULL	NULL

2.7 Type the SQL query that returns all employees who earn 45,000 or more. Order the results from highest earner to lowest

```
SELECT firstname, lastname, departmentid
FROM employee
WHERE salary >= 45000
ORDER by salary DESC;
```

Your result set should look like this

firstname	lastname	salary	departmentid
Alice	Munro	125000.00	1
Sarah	Fergusson	86000.00	9
Ned	Kelly	85000.00	11
Sophie	Monk	75000.00	10
Brier	Patch	73000.00	9
Todd	Beamer	68000.00	8
Andrew	Jackson	55000.00	11
Clare	Underwood	52000.00	11
Nancy	Cartwright	52000.00	8
Gigi	Montez	46000.00	3
Maggie	Smith	46000.00	3
Sanjay	Patel	45000.00	6
Rita	Skeeter	45000.00	2
James	Mason	45000.00	4
Pat	Clarkson	45000.00	5
Mark	Zhang	45000.00	7

2.8 Type the SQL query that returns all rows and columns in the employee table. Order the result set by departmentid then alphabetically by employee's lastname.

```
SELECT *
FROM employee
ORDER BY departmentid, lastname;
```

EmployeeID	FirstName	LastName	Salary	DepartmentID	BossID	DateOfBirth
1	Alice	Munro	125000.00	1	NULL	1966-12-14
11	Rita	Skeeter	45000.00	2	4	1988-02-22
12	Gigi	Montez	46000.00	3	4	1992-03-20
13	Maggie	Smith	46000.00	3	4	1991-04-29
14	Paul	Innit	41000.00	4	3	1998-06-02
15	James	Mason	45000.00	4	3	1995-07-30
16	Pat	Clarkson	45000.00	5	3	1997-08-28
10	Saniav	Patel	45000.00	6	3	1984-01-28
17	Mark	Zhang	45000.00	7	3	1996-10-01
5	Todd	Beamer	68000.00	8	1	1965-05-24
6	Nancy	Cartwright	52000.00	8	5	1993-04-11
8	Sarah	Fergusson	86000.00	9	7	1978-11-15
7	Brier	Patch	73000.00	9	1	1981-10-16
9	Sophie	Monk	75000.00	10	1	1986-12-15
3	Andrew	Jackson	55000.00	11	2	1958-04-01
2	Ned	Kelly	85000.00	11	1	1970-07-16
4	Clare	Underwood	52000.00	11	2	1982-09-22

Limit

We can limit the number of rows in the result set by using the word LIMIT and specifying an integer after the LIMIT word.

```
SELECT Name
FROM department
WHERE Floor = 5
ORDER BY Name ASC
LIMIT 2;
```

Name
Accounting
Management

2.9 Type the above query and note the two rows returned. Change the ORDER BY from ASC to DESC and rerun the query. Is the result set different? If so, why are they different? [Hint: remove the LIMIT].

2.10 Type the SQL query :that returns the first name, last name and salary of the five highest salary earners across the whole Department store.

Your result set should look like this

firstname	lastname	salary
Alice	Munro	125000.00
Sarah	Fergusson	86000.00
Ned	Kelly	85000.00
Sophie	Monk	75000.00
Brier	Patch	73000.00

END OF Week 3 LAB

Appendix – Department Store conceptual design