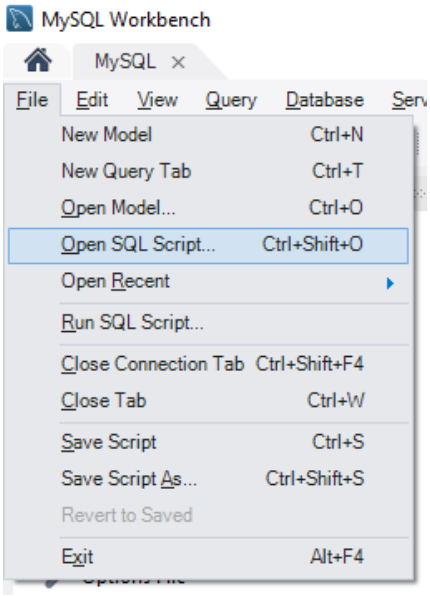
INFO90002 Tutorial Week 6 SOLUTIONS

Objectives

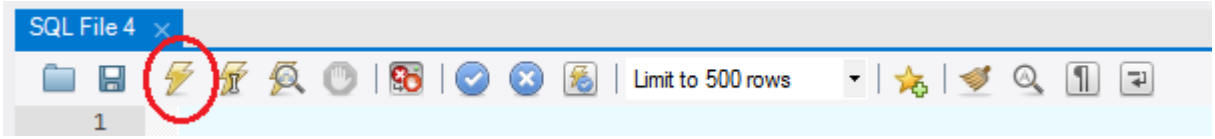
- Learn how to type SQL SELECT commands
- Learn how to FILTER by row and column.
- Learn to use maths conditions
- Learn how to use ORDER BY, LIMIT
- Learn how to use FUNCTIONS (max, min, sum, avg)
- Learn how to use GROUP BY

1) Using MySQL Workbench, connect to your database to the MySQL Server
2) **TASK** Download the labs2018v6.sql script from the LMS
3) **TASK** From the SQL Query window Choose File -> Open SQL Script



Select the place where you downloaded the labs2018v6.sql script

4) **TASK** Click the lightening icon to execute



5) **TASK** Show the catalogue (meta data) 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 1 you may also need to tell the database to use the correct schema

```
use <username>;
```

SELECT

SELECT statements retrieve and display data from the database. 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
ORDER BY (grouping list) -- optional
LIMIT (number of rows) OFFSET (number of rows) -- 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

6) **TASK** 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

```
SELECT *
FROM employee;
```

Filtering

To select only some columns (projection in Relational Algebra), 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.

```
DESC department;
```

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.

7) **TASK** 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
Saniay	Patel	6
Rita	Skeeter	2
Gigi	Montez	3
Maggie	Smith	3
Paul	Innit	4
James	Mason	4
Pat	Clarkson	5
Mark	Zhang	7

```
SELECT firstname, lastname, departmentid
FROM employee;
```

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* in Relational Algebra. 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

8) **TASK** 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;
```

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

9) **TASK** In MySQL Workbench show the metadata about the Department and Employee tables

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

Maths Conditions

The table below summarises the conditions and their interpretation in SQL:

Condition	Meaning	Example	Explanation
>	Greater Than	salary > 55000	value is 55001 or higher
<	Less Than	salary < 45000	value is 44999 or lower
=	Equal to	lastname = 'Underwood'	must match exactly
>=	Greater than or equal to	salary >= 450000	value is 45000 or higher
<=	Less than or equal to	salary <= 450000	value is 45000 or lower
!= <>	Not equal to	name != 'Torch' name <> 'Torch'	can be any value except Torch

Table 1: Conditions and their behaviour in SQL

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

10) **TASK** Type the SQL query that lists the first name, last name, departmentid and salary of all employees who work in DepartmentID 11 AND who earn greater than 55000.

```
SELECT firstname, lastname
FROM employee
WHERE departmentid=11
AND salary > 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

11) **TASK** 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
Saniav	Patel	45000.00	6
Rita	Skeeter	45000.00	2
James	Mason	45000.00	4
Pat	Clarkson	45000.00	5
Mark	Zhang	45000.00	7

12) **TASK** 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

13) TASK 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: to see the diference remove the LIMIT.

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

```
SELECT firstname, lastname, salary
FROM employee
ORDER BY salary DESC
LIMIT 5:
```

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

FUNCTIONS

Functions are mathematical and scientific calculations that are performed automatically by the database engine. There are several function types across all database data types. The most common functions we use are COUNT, MAX, MIN. The full list of Functions you can use in MYSQL are found here in the MySQL reference manual

To find out how many departments there are we can use the COUNT() function. Functions must be given something to act on which can be a column, or all columns using the wild card *

E.G.

```
SELECT COUNT(*)
FROM Department;
```

	COUNT(*)
	11

```
SELECT COUNT(Name)
FROM Department;
```

	COUNT(name)
	11

```
SELECT CONCAT(FirstName,' ',LastName , ' works in the ' ,
Department.Name, ' Department') AS INFO
FROM EMPLOYEE NATURAL JOIN DEPARTMENT;
```

*Note we did a **join** between two tables Employee and Department*

More about joining tables in the next lab.

	INFO
	Alice Munro works in the Management Department
	Rita Skeeter works in the Books Department
	Gigi Montez works in the Clothes Department
	Maggie Smith works in the Clothes Department
	Paul Innit works in the Equipment Department
	James Mason works in the Equipment Department
	Pat Clarkson works in the Furniture Department
	Sanjay Patel works in the Navigation Department
	Mark Zhang works in the Recreation Department
	Todd Beamer works in the Accounting Department
	Nancy Cartwright works in the Accounting Department
	Brier Patch works in the Purchasing Department
	Sarah Fergusson works in the Purchasing Department
	Sophie Monk works in the Personnel Department
	Ned Kelly works in the Marketing Department
	Andrew Jackson works in the Marketing Department
	Clare Underwood works in the Marketing Department

15) **TASK** Type the SQL query to find the total number of employees in the employee table

Your result set should look like this

	count(*)
▶	17

```
SELECT COUNT(*)
FROM Employee;
```

Alternatively:

```
Select count(lastname)
FROM Employee;
```

GROUP BY

Sometimes we want to group the function by a particular attribute. For example to find out the number of each departments on each floor of the department store we would type:

```
SELECT floor, count(departmentid)
FROM DEPARTMENT
GROUP BY floor;
```

floor	COUNT(departmentid)
1	2
2	2
3	1
4	1
5	5

We use the GROUP BY keyword when aggregate functions are with a column that does not aggregate the rows. We must group by the non aggregated column or columns to ensure the full result set is returned. Thus in the above example we GROUP BY *floor.*

Try this: Remove the GROUP BY keyword and notice the difference in the query output

Alias

We can also alias the columns to make the output make more sense to the reader. Then use that alias within the query

```
SELECT floor as DEPT_FLOOR, COUNT(departmentid) AS DEPT_COUNT
FROM DEPARTMENT
GROUP BY DEPT_FLOOR
ORDER BY DEPT_FLOOR;
```

DEPT_FLOOR	DEPT_COUNT
1	2
2	2
3	1
4	1
5	5

16) **TASK** Type the SQL query to find how many employees work in each department

departmentid	count(employeeid)
1	1
2	1
3	2
4	2
5	1
6	1
7	1
8	2
9	2
10	1
11	3

```
SELECT DepartmentID, Count(EmployeeID)
FROM Employee
GROUP BY DepartmentID;
```

17) **TASK** Type the SQL query to find each department's average salary?

departmentid	avg(salary)
1	125000.000000
2	45000.000000
3	46000.000000
4	43000.000000
5	45000.000000
6	45000.000000
7	45000.000000
8	60000.000000
9	79500.000000
10	75000.000000
11	64000.000000

```
SELECT DepartmentID, AVG(Salary)
FROM Employee
GROUP BY DepartmentID;
```

18) **TASK** Type the SQL query that finds what department has the highest salary?

DepartmentID	MAX(Salary)
1	125000.00

```
SELECT DepartmentID, MAX(Salary)
FROM Employee
Group By DepartmentID
ORDER BY MAX(Salary) DESC
LIMIT 1;
```

19) **TASK** Type the SQL query that finds the department with the lowest salary?

DepartmentID	MIN(Salary)
4	41000.00

```
SELECT DepartmentID, MIN(Salary)
FROM Employee
Group By DepartmentID
ORDER BY MIN(Salary)
LIMIT 1;
```

END OF WORKSHOP

Appendix New department Store Physical ER Model

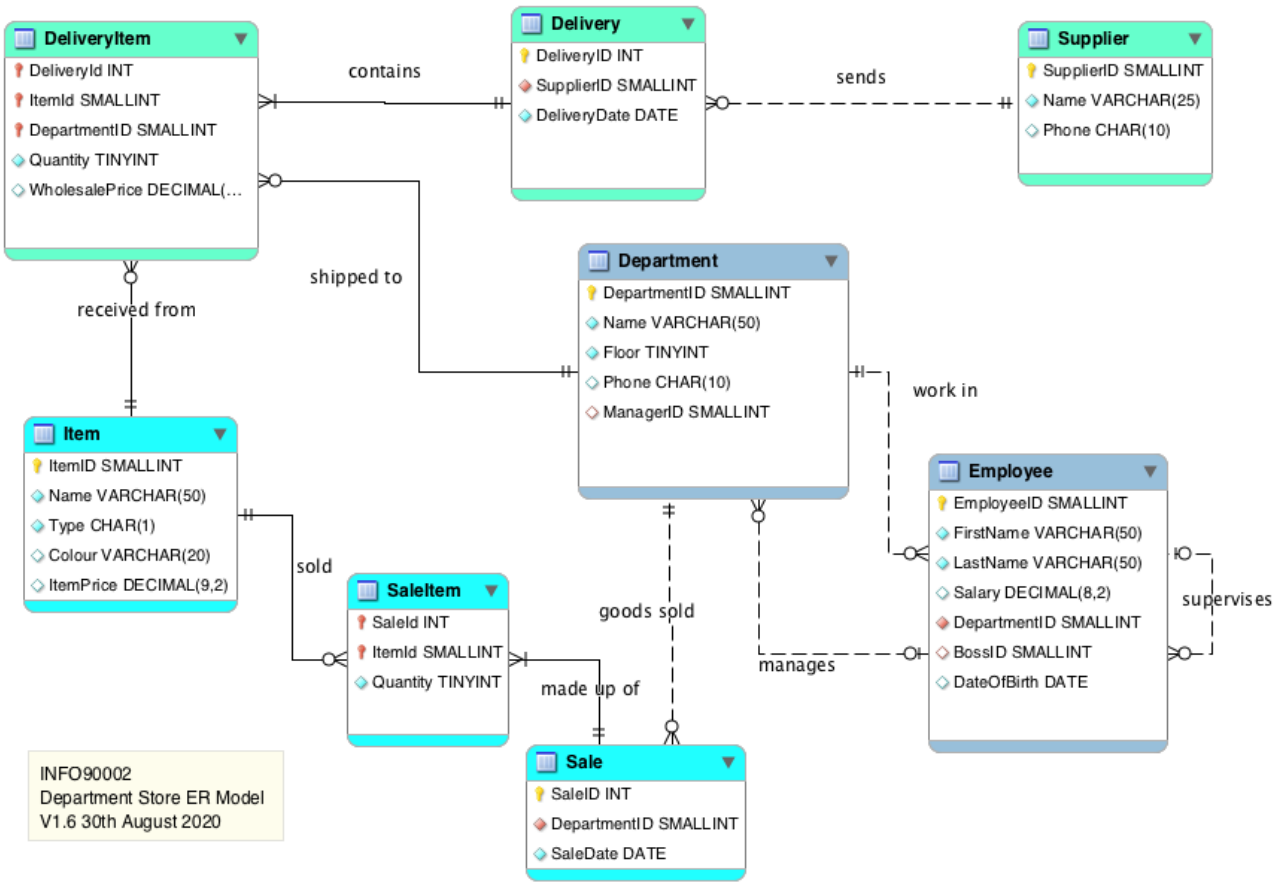


Figure A1: The department store schema