INFO90002 Week 4 - ANSWERS

Objectives:

- Finalise a physical model of the Department Store database
- Learn SQL by example

Section 1. Data Modelling

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, 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.

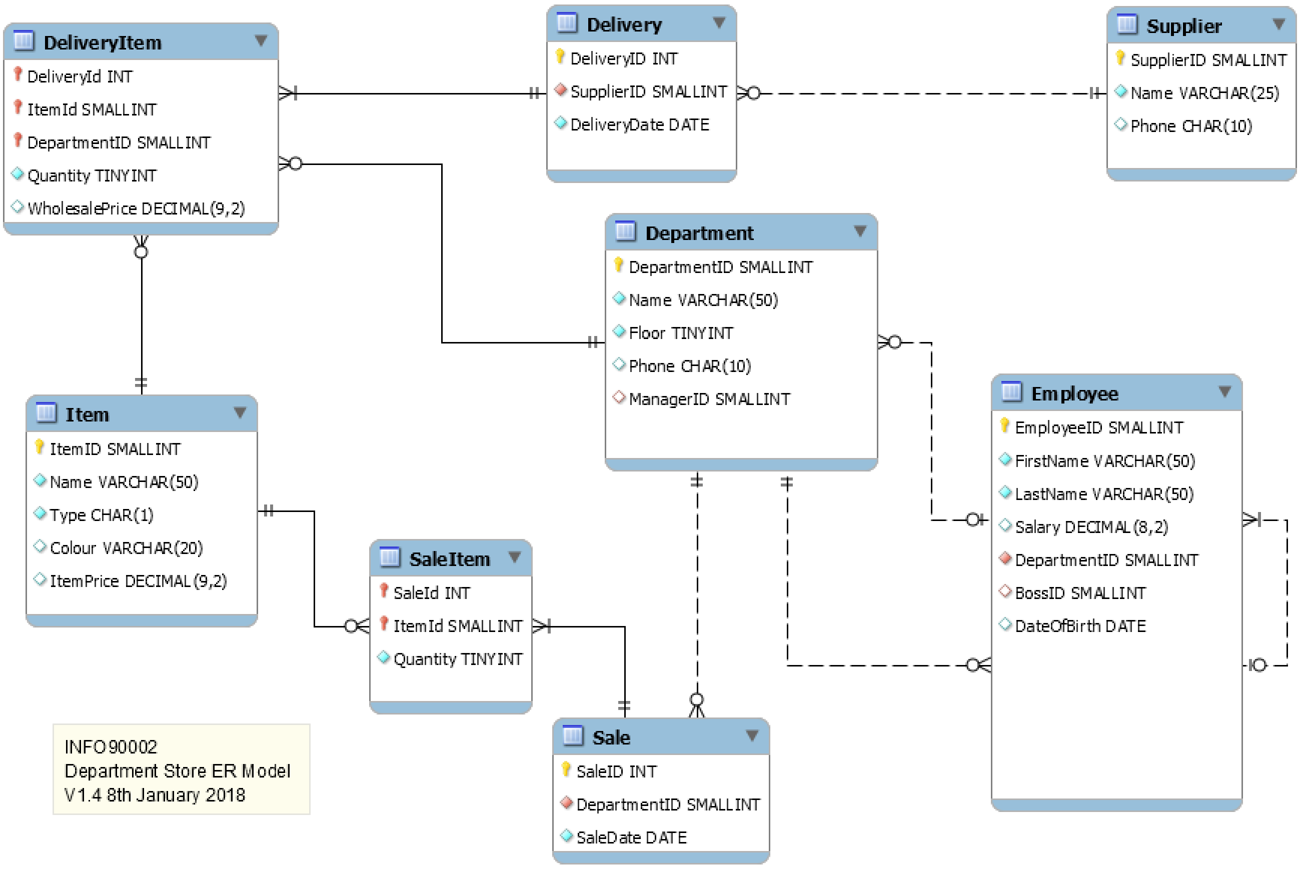
Items are delivered to the store by suppliers. Each delivery from a supplier contains one or more items delivered to one or more departments within the store. We record which supplier made the delivery and to which department, and the wholesale price of each item. For each supplier we record a unique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

1.1. Review the case study, your pen and paper conceptual data model from Lab 1 and your MySQL Workbench logical model from Lab 2. In MySQL Workbench now design a physical model for a MySQL relational database.

> *HINT: When designing a physical model you should choose datatypes available to that particular database engine. Review the MySQL datatypes available to you and be sure to choose the appropriate data type and length for each column.*

The following section you will type and copy SQL against the lab data set. The lab data is a model of the Department Store. It is always handy to use the ER model to understand the relationship between entities.

Physical Data Model: The Department Store



Section 2 Learning SQL by example

2.1 List the first name, last name and salary of the five highest salary earners across the Department store.

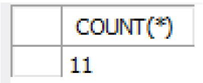
```
SELECT FirstName, LastName, Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 5;
```

OPERATORS and 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

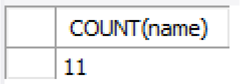
To find out how many departments there are we can use the COUNT() function. Functions must be given something to act on which can be a column, or all columns using the wild card *
E.G.

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



	COUNT(*)
	11

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

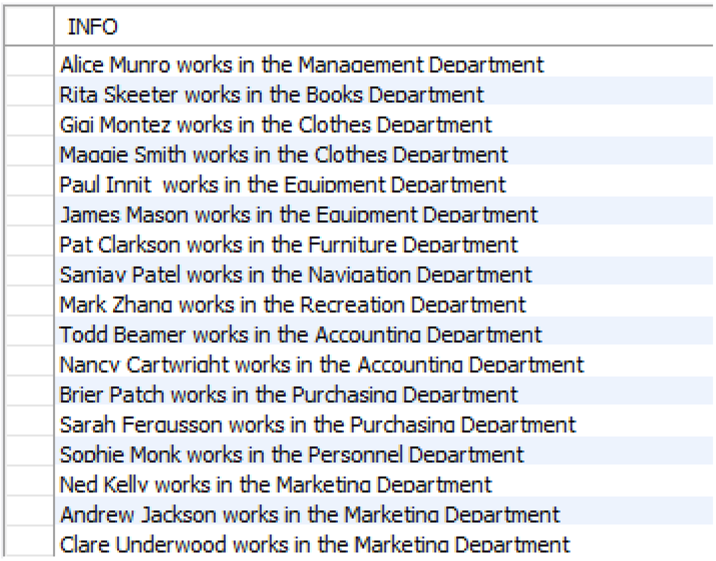


	COUNT(name)
	11

```
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*

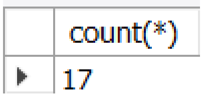
More about joining tables in the next lab.



	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

2.2 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;
```

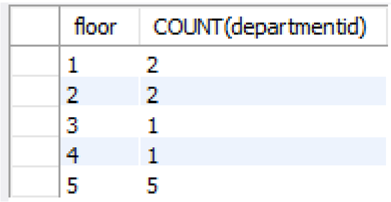
Alternatively:

```
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

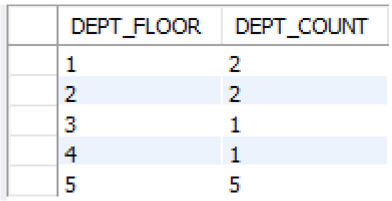
We use the GROUP BY keyword when aggregate functions are with a column that does not aggregate the rows. We must group by the non aggregated column or columns to ensure the full result set is returned. Thus in the above example we GROUP BY *floor*.

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

Alias

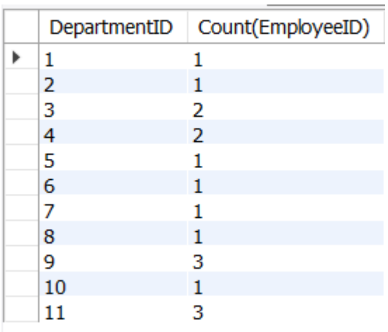
We can also alias the columns to make the output make more sense to the reader. Then use that alias within the query

```
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

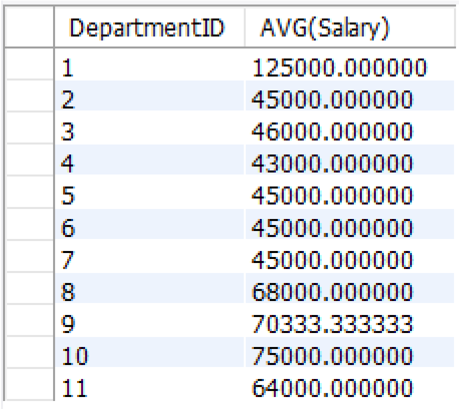
2.3 Type the SQL query to find how many employees work in each department



DepartmentID	Count(EmployeeID)
1	1
2	1
3	2
4	2
5	1
6	1
7	1
8	1
9	3
10	1
11	3

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

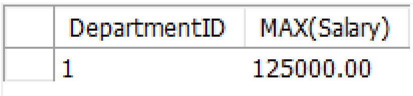
2.4 Type the SQL query to find each department’s average salary?



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	68000.000000
9	70333.333333
10	75000.000000
11	64000.000000

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

2.5 Type the SQL query that finds what department has the highest salary?

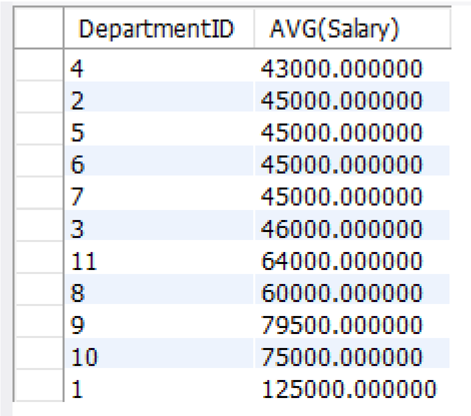


DepartmentID	MAX(Salary)
1	125000.00

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

2.6 Type the SQL query that finds the department with the lowest average salary?

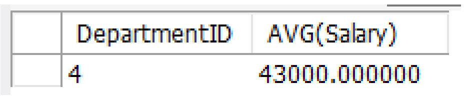
```
SELECT DepartmentID, AVG(Salary)
FROM Employee
Group By DepartmentID
ORDER BY MIN(Salary)
```



DepartmentID	AVG(Salary)
4	43000.000000
2	45000.000000
5	45000.000000
6	45000.000000
7	45000.000000
3	46000.000000
11	64000.000000
8	60000.000000
9	79500.000000
10	75000.000000
1	125000.000000

Now we can limit the rows as there is only 1 department with the lowest salary

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



DepartmentID	AVG(Salary)
4	43000.000000

Formatting & Rounding your results

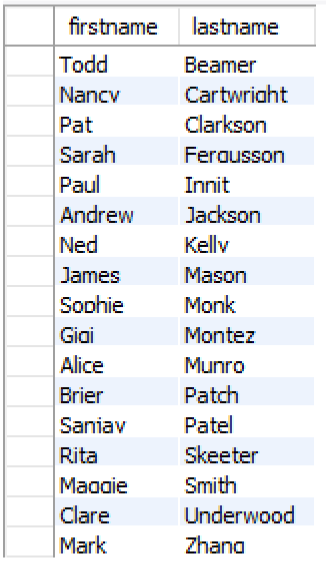
FORMAT(X,D) and ROUND(X,D) are functions you can use to improve the readability of a query result. Round will round the argument – what is in the parenthesis “X” to D decimal places. Format will format the argument to D decimal places and include commas.

```
SELECT AVG(Salary) AS AVG_SAL
FROM Employee;

60529.411765

SELECT FORMAT(AVG(SALARY),2) AS AVG_SAL
FROM Employee;

60,529.41
SELECT ROUND(AVG(SALARY),2) AS AVG_SAL
FROM Employee;
60529.41
```

Format and Round while producing the same output are subtly different. Format converts the output into a STRING (hence the 60<comma>529), whereas round keeps the result as a NUMBER

2.7 Find the first and last names of all the employees

```
SELECT firstname, lastname
FROM employee
ORDER BY lastname;
```



firstname	lastname
Todd	Beamer
Nancy	Cartwright
Pat	Clarkson
Sarah	Fergusson
Paul	Innit
Andrew	Jackson
Ned	Kelly
James	Mason
Sophie	Monk
Gigi	Montez
Alice	Munro
Brier	Patch
Sanjay	Patel
Rita	Skeeter
Maggie	Smith
Clare	Underwood
Mark	Zhang

2.8 List each employee's full name and the department name they work in. Order the result by department name

This query requires you to join the Department table to the Employee table. Use the Physical data model to work out if you need to do a natural join or an inner join

```
SELECT name, firstname, lastname
FROM department natural join employee
ORDER by name;
```

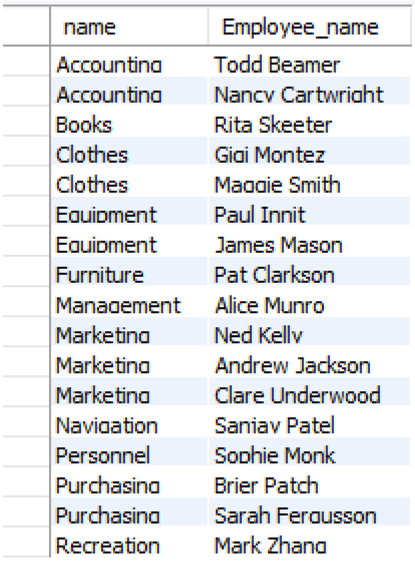
Alternatively you could format the query for better readability

```
SELECT name as Department_name,concat(firstname, ' ',lastname) as
Employee_name
FROM department natural join employee
ORDER by name;
```

Alternatively using an inner join

```
SELECT name as Department_name,concat(firstname, ' ',lastname) as
Employee_name
```

```
FROM department inner join employee
ON department.DepartmentID = employee.DepartmentID
ORDER by name;
```

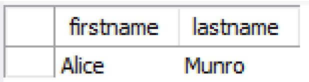


name	Employee_name
Accounting	Todd Beamer
Accounting	Nancy Cartwright
Books	Rita Skeeter
Clothes	Gigi Montez
Clothes	Maggie Smith
Equipment	Paul Innit
Equipment	James Mason
Furniture	Pat Clarkson
Management	Alice Munro
Marketing	Ned Kelly
Marketing	Andrew Jackson
Marketing	Clare Underwood
Navigation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

2.9 Type a query to find the first and last name of all the employees in department 1. Then test your written SQL in MySQL Workbench

```
SELECT firstname, lastname
FROM employee
WHERE departmentid=1;
```

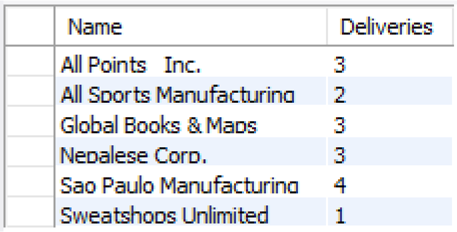
Your result set should look like this:



firstname	lastname
Alice	Munro

2.10 List the Supplier name and the number of deliveries made to the department store

```
SELECT Supplier.Name, count(Delivery.DeliveryID) as numDeliveries
FROM supplier natural join delivery
GROUP by supplier.Name;
```



Name	Deliveries
All Points Inc.	3
All Sports Manufacturing	2
Global Books & Maps	3
Nepalese Corp.	3
Sao Paulo Manufacturing	4
Sweatshops Unlimited	1

In this SQL we have prefaced each column with its table name for readability. This format is always

```
TABLENAME.COLUMNNAME
```

END of LAB