INFO20003 Week 6 Lab

Objectives:

- Learn unary and outer joins in SQL
- Self-test your SQL skills

Section 1: Continuing SQL

Unary joins

Unary joins are a type of join where the table is joined to itself. Within our Department store schema, employees can be bosses of other employees. Sometimes we will need to find information about each employee's boss:

```
SELECT
  emp.FirstName AS employee_first,  -- employee first name
  emp.LastName AS employee_last,  -- employee last name
  emp.departmentID,  -- employee department id
  boss.FirstName AS boss_first,  -- boss first name
  boss.LastName AS boss_last  -- boss last name
FROM employee AS emp INNER JOIN employee AS boss
  ON emp.BossID = boss.employeeID
  -- join the boss ID in emp to employee ID in boss
ORDER BY departmentID, employee_last;
```


Inspect the FROM clause. The employee table is first aliased as *emp*, then listed again aliased as *boss*. To the database server this is now effectively two tables: *emp* and *boss*. One table list the employees (subordinates), the other table lists the bosses of those employees.

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

This query only returns sixteen rows. However, we have seventeen employees. Alice Munro does not have a boss, so her record is missing as an employee. We can fix this by using a LEFT JOIN, which is a type of outer join.

Outer joins

Outer joins are used when we wish to display all records in one table even if there is no matching row in the other table.

In the above example, the *emp* copy of the employee table has no boss for Alice Munro – the value for BossID for Alice Munro is NULL. Therefore the join condition `ON emp.bossID = boss.employeeID` does not have a match. There is no boss with an employeeID of NULL. (Nor can there be, as employeeID is a primary key – primary key columns can never be NULL.) Because there is no match, Alice Munro's row is not part of the result set.

An outer join resolves this by returning rows that do not have a match in the Boss table. MySQL Server supports the use of LEFT JOIN and RIGHT JOIN. This query uses LEFT JOIN:

```
SELECT
  emp.FirstName AS employee_first,
  emp.LastName AS employee_last,
  emp.departmentID,
  boss.FirstName AS boss_first,
  boss.LastName AS boss_last
FROM employee AS emp LEFT JOIN employee AS boss
  ON emp.BossID = boss.employeeID
ORDER BY departmentID, employee_last;
```


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

Alice Munro is now listed as an employee with no boss, and NULL is returned as the boss first and last name.

In the above example using a LEFT JOIN, all records on the *left* table are kept, irrespective of whether a match exists in the right table. Whether to use a LEFT JOIN or a RIGHT JOIN is dependent on which table's records must be completely preserved. Here is the same query written as a RIGHT JOIN:

```
SELECT
  emp.FirstName AS employee_first,
  emp.LastName AS employee_last,
  emp.departmentID,
  boss.FirstName AS boss_first,
```

```
  boss.LastName AS boss_last
FROM employee AS boss RIGHT JOIN employee AS emp
  ON emp.BossID = boss.employeeID
ORDER BY departmentID, employee_last;
```

The result set is the same as before.

Practicing unary joins

- **Task 1.1** Find the names of employees who work in the same department as their boss. Report the full name of the employee, the department, and the boss's name. *Hint: Fill in the gaps in the following query:*

```
SELECT ...
FROM employee AS emp INNER JOIN employee AS boss
  ON emp.BossID = boss.employeeID
WHERE emp.departmentID = ...
```

Your result set should look like this:

	employee_name	departmentID	boss_name
▸	Andrew Jackson	11	Ned Kelly
	Clare Underwood	11	Ned Kelly
	Nancy Cartwright	8	Todd Beamer
	Sarah Fergusson	9	Brier Patch

- **Task 1.2** List the IDs of the departments where all the employees earn less than their boss.

```
SELECT DISTINCT departmentID
FROM employee
WHERE departmentID NOT IN
     (SELECT emp.departmentID
      FROM employee AS emp INNER JOIN employee AS boss
        ON emp.BossID = boss.employeeID
      WHERE emp.Salary >= boss.Salary);
```

Notice that the inner query uses a unary join to create a result set that lists all departmentIDs where at least one employee earns more than their boss. That is why the condition is NOT IN.

	DepartmentID
	1
	2
	3
	4
	5
	6
	7
	8
	10
	11

◆ **Task 1.3** Type a query to find the name, salary, and boss's name of the employees of department ID 11 who have a salary over $55,000.
Hint: This will require a unary join to find the bosses and employees. You will need to provide a WHERE clause to fulfil the two conditions listed.

Your result set should look like this:

	EmployeeName	Salary	Manager
	Ned Kelly	85000.00	Alice Munro

Bringing together the concepts so far

SQL queries are provided for some of the following tasks – try to write the query on your own before trying the suggested query for yourself in Workbench.

◆ **Task 1.4** Find the items delivered by at least two suppliers.

```
SELECT item.Name
FROM item NATURAL JOIN deliveryitem NATURAL JOIN delivery
GROUP BY item.Name
HAVING COUNT(DISTINCT SupplierID) >= 2;
```


COUNT(SupplierID) will count the rows per item for which SupplierID is not NULL. If we want to count the number of *different* suppliers per item, we need to count the number of *distinct* SupplierID values.

	Name
▶	Compass - Silva
	Exploring in 10 Easy Lessons
	Geo positioning system
	Gortex Rain Coat
	How to Win Foreign Friends
	Map case
	Map measure
	Pocket knife - Essential
	Pocket knife - Steadfast
	Torch

◆ **Task 1.5** Type a query to return the items that have been sold by at least two departments.

	Name
▶	Compass - Silva
	Geo positioning system
	Gortex Rain Coat
	How to Win Foreign Friends
	Pocket knife - Essential
	Torch

◆ **Task 1.6** Find the name of the highest-paid employee in the Marketing department.

```
SELECT employee.Firstname, employee.Lastname, employee.Salary
FROM employee NATURAL JOIN department
WHERE department.Name = 'Marketing'
  AND employee.Salary =
     (SELECT MAX(salary)
      FROM employee NATURAL JOIN department
      WHERE department.Name = 'Marketing');
```

◆ **Task 1.7** Find the names of employees who earn at least 30 per cent less than the average salary.

```
SELECT firstname, lastname, salary
FROM employee
WHERE salary <
      (SELECT AVG(salary) * 0.7
       FROM employee);
```

◆ **Task 1.8** Find the number of employees with a salary under $45,000.

```
SELECT COUNT(*)
FROM employee
WHERE salary < 45000;
```


◆ **Task 1.9** Find the number of units sold of each item.
Hint: The "number of units sold" is the total quantity of that item that has been sold.


```
SELECT item.Name, SUM(saleitem.quantity) AS UnitsSold
FROM saleitem NATURAL JOIN item
GROUP BY ItemID
ORDER BY Name;
```


This query does not return the fact that the Horse Saddle (ItemID 1) has never been sold! As such, to retrieve all of the information, you need to use an outer join:

```
SELECT item.Name, SUM(saleitem.quantity) AS UnitsSold
FROM saleitem
  RIGHT JOIN item ON saleitem.ItemID = item.ItemID
GROUP BY item.ItemID
ORDER BY item.Name;
```

When solving queries of this type, always consider the possibility that some items may have never been sold, delivered, etc.

◆ **Task 1.10** Find any suppliers that have delivered no more than two unique items. List the suppliers in alphabetical order.

```
SELECT supplier.Name
FROM delivery INNER JOIN supplier INNER JOIN deliveryItem
  ON supplier.SupplierID = delivery.SupplierID
  AND delivery.DeliveryID = deliveryitem.DeliveryID
GROUP BY supplier.SupplierID
HAVING COUNT(DISTINCT deliveryitem.ItemID) <= 2
ORDER BY supplier.Name;
```

◆ **Task 1.11** Find the names of suppliers that have never delivered a compass.

```
SELECT DISTINCT supplier.Name
FROM supplier
WHERE supplier.SupplierID NOT IN
    (SELECT SupplierID
     FROM delivery NATURAL JOIN deliveryitem NATURAL JOIN item
     WHERE item.Name LIKE 'Compass%');
```

This question was challenging in that you were not given the complete data. The Item name for the Compass in the database is 'Compass - Silva' but you only had part of the information. In this case you cannot get an exact match, so a condition such as `= 'Compass'` would return no rows. To find rows that match, you must use the LIKE condition and the wildcard %.

This is why it is always good to query reference tables like Item to know how the item name is actually stored in the database.

Section 2: SQL self-test: single-table queries

Note: You will need to read the functions section of the MySQL reference manual to answer some of these questions: https://dev.mysql.com/doc/refman/8.0/en/functions.html

◆ **Task 2.1** How many deliveries have there been in the month of July?
Hint: The only information you have been given is the month name.

	count(deliverydate)
	3

◆ **Task 2.2** List the names of the tents available for sale.

	name
▶	Tent - 2 person
	Tent - 8 person
	Tent - 4 person

◆ **Task 2.3** What month has had the highest number of sales?

	month	num_sales
▶	October	21

◆ **Task 2.4** List the salary total and employee count for each departmentID. Order the results from the smallest salary total to the largest.

departmentid	SUM(salary)	COUNT(*)
5	45000.00	1
6	45000.00	1
2	45000.00	1
7	45000.00	1
10	75000.00	1
4	86000.00	2
3	92000.00	2
8	120000.00	2
1	125000.00	1
9	159000.00	2
11	192000.00	3

◆ **Task 2.5** How many sales have been on a Sunday?

count(saleid)
6

◆ **Task 2.6** How many days have elapsed between the first delivery date and most recent delivery date for each supplier?

supplierid	datediff
101	165
102	121
103	66
104	0
105	110
106	88

◆ **Task 2.7** Produce the following output by writing a SQL statement.

Where is each department?
The Accounting department is on floor number 5
The Books department is on floor number 1
The Clothes department is on floor number 2
The Equipment department is on floor number 3
The Furniture department is on floor number 4
The Management department is on floor number 5
The Marketing department is on floor number 5
The Navigation department is on floor number 1
The Personnel department is on floor number 5
The Purchasing department is on floor number 5
The Recreation department is on floor number 2

◆ **Task 2.8** Find the minimum, maximum, average and standard deviation for salaries in each department.

departmentid	MIN	MAX	STDDEV
1	125000.00	125000.00	0
2	45000.00	45000.00	0
3	46000.00	46000.00	0
4	41000.00	45000.00	2000
5	45000.00	45000.00	0
6	45000.00	45000.00	0
7	45000.00	45000.00	0
8	52000.00	68000.00	8000
9	73000.00	86000.00	6500
10	75000.00	75000.00	0
11	52000.00	85000.00	14899.66442575134

◆ **Task 2.9** List the green items of type C.

ItemID	Name
12	Gortex Rain Coat

End of Week 6 Lab