

Tutorial - Week 6

Objectives

- Learn how to use the following SQL keywords: FORMAT, ROUND
- Learn how to join two tables together with INNER JOIN and NATURAL JOIN
- Learn how to write sub queries
- Learn to use maths operators
- Learn how to use SQL keywords: HAVING, UNION

Formatting & Rounding

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_Salary
FROM Employee;
```

```
60529.411765
```

```
SELECT FORMAT(AVG(Salary),2) AS Avg_Salary
FROM Employee;
```

```
60,529.41
```

```
SELECT ROUND(AVG(Salary),2) AS Avg_Salary
FROM Employee;
60529.41
```

N.B.: FORMAT and ROUND while producing the same output are different. FORMAT converts the output into a STRING (hence the 60<comma>529), whereas round keeps the result as a NUMBER

NATURAL JOINS and INNER JOINS

Typically, relational database management systems have many entities in a database schema. To retrieve all the necessary data, tables frequently need to be joined together. Table joins can take many forms in SQL. They include Natural Join, Inner Join, Right Join, Left Join, Outer Join. We will now look at the Natural and Inner joins in SQL.

NATURAL JOIN

Natural Joins work when the joining columns names are identical in both tables that are being joined together.

```
SELECT table1.column1, table1.column2, table2.column1
FROM table1
NATURAL JOIN table2
```

Consider the department table:

PK

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

And the employee table:

FK

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	Ferrousion	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	Gia	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
NULL	NULL	NULL	NULL	NULL	NULL	NULL

The departmentID column is common to both tables. There is a foreign key (departmentID) in the Employee table that references the primary key (departmentID) in the Department table.

To find the department name for each employee we would need to retrieve the name from the Department table and firstname and lastname from the Employee table (joined over departmentID).

```
SELECT department.name, employee.firstname, employee.lastname
FROM employee
NATURAL JOIN department;
```

The result set would be:

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

It is ok to do this ↴

```
SELECT department.name, employee.firstname, employee.lastname
FROM department
NATURAL JOIN employee;
```

Because each column name in the SELECT command is unique to the query, the department name and employee names could also be written without the table name prefacing the column name:

```
SELECT name, firstname, lastname
FROM employee
NATURAL JOIN department;
```

Natural Joins work when the column name is identical in BOTH name and purpose for the two tables being joined together. However, ***there are times when the column names are identical - but the meaning is different, and they refer to different characteristics.***

For example, both the Department and Item tables have a column called name. But the purpose is different. The name column in the Department table is the department's name and the name column in the Item table is the item's name. Yet despite there being no PK-FK relationship between the Item and Department tables the following statement is parsed and executed.

```
SELECT itemid, departmentID
FROM item
NATURAL JOIN department;
```

It returns 0 rows because there are no common names in the respective name columns of the item and department table. The MySQL database server does **not** return an error because the SQL is *syntactically* correct although it is logically incorrect.

INNER JOIN

Inner Joins **are always used when the joining column names are not identical**. Inner Joins provide an explicit definition of what the join condition is. INNER JOIN must always be joined using the **ON syntax which specifies the join condition**.

Syntax for an INNER JOIN is the following (note the ON clause):

```
SELECT table1.column1, table1.column2, table2.column2
FROM table1
INNER JOIN table2
ON table1.column1 = table2.column4;
```

Inspect the earlier Department / Employee name query now rewritten as an Inner Join:

```
SELECT name, firstname, lastname
FROM department
INNER JOIN employee
ON department.departmentID = employee.departmentID;
```

The INNER JOIN syntax explicitly states what column should join the two tables, whereas the NATURAL JOIN is implicit. The query using an Inner Join explicitly states the join is on the departmentID column in both the department and employee tables.

- 1) **TASK.** 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	Gia 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	Saniav Patel
Personnel	Sophie Monk
Purchasing	Brier Patch
Purchasing	Sarah Fergusson
Recreation	Mark Zhang

Note, in most cases INNER JOIN executes faster than NATURAL JOIN, therefore we do not recommend using NATURAL JOIN.

- 2) **TASK** Type a query to find the first and last name of all the employees in the Management department. Then test your written SQL in MySQL Workbench

Your result set should look like this:

firstname	lastname
Alice	Munro

```
select firstname, lastname
from employee
where department_id = 10
```

- 3) **TASK** Type the query to list the Supplier name and the number of deliveries made to the department store

```
where department_id = 10
```

Your result set should look like this:

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


Sub Queries

Sometimes we need to find a value and then use it. For example if we wanted to find out the departmentid of the department which has the lowest salary.

First we would find the lowest salary across all of the department store.

You might be tempted to write a query like this

```
SELECT MIN(Salary), DepartmentID
FROM Employee;
```

But it will not run. MySQL will generate an error.  when we use an aggregation function such as MIN(), any non-aggregated attribute in the SELECT clause must also appear in a GROUP BY clause. The correct version of the query is:

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

The result set will show that the department with lowest salary is department 4, where an employee has a salary of \$41,000.

While we can ORDER BY the result in descending order of MIN(salary) and then use the LIMIT keyword – it is not wise to do so and is to be avoided if possible.

We can break the problem into two components. Find the lowest salary, then find the departmentID rows where the salary matches the value of the first query.

```
SELECT MIN(Salary)
FROM Employee;
```

The result set is one value {41000}, we then use that value in our next query

```
SELECT DepartmentID
FROM Employee
WHERE Salary = 41000;
```

This can be done in the one statement:

```
SELECT departmentid
FROM Employee
WHERE SALARY =
    (SELECT MIN(Salary)
     FROM Employee);
```

	departmentid
	4

The query in parenthesis is known as the "inner query" and the other query is known as the "outer query".

This is doing exactly the same as the two individual SQL statements above it but in one statement. The query in parenthesis is run first and the result returned (41000). Then each row in the outer query evaluates the salary value with the value returned in the inner query (41000). The only match is where Department ID is 4.

important: if the subquery returns more than one result "=" will not work you must use "IN"

4) How many employees work in departments located on the fifth floor?

To approach this question, we break it into separate components.

1. What departmentid's are on the fifth floor?
2. Count the number of employees whose departmentid matches the results returned in the inner query who are in the same department as the first query

Query 1 The "inner" query

```
SELECT departmentid
FROM Department
WHERE Floor = 5;
```

more than one result

departmentid
1
8
9
10
11

Returns the result set {1,8,9,10,11} that is five rows.

Now we need to count each row in the Employee table where the departmentID matches 1 or 8 or 9 or 10 or 11.

As the result set has more than one row we need to use the keyword IN

```
SELECT COUNT(employeeid)
FROM Employee
WHERE departmentid IN
  (SELECT departmentID
   FROM Department
   WHERE Floor = 5);
```

EMP_COUNT_FLOOR5
9

TRY: If you have time replace the keyword IN with "=" and observe the error in the query window:
Error Code: 1242. Subquery returns more than 1 row

Multiple table joins

When we join more than two tables, the same principles apply, there must be a common column between the two entities functioning as a Primary Key / Foreign Key referential integrity.

5) TASK Find the sale dates of all types of tents

Using an INNER JOIN

	name	saledate	SUM(saleitem.quantity)
▶	Tent - 2 person	2017-10-14	1
	Tent - 2 person	2017-10-15	1
	Tent - 2 person	2017-10-24	1
	Tent - 2 person	2017-10-25	2
	Tent - 8 person	2017-08-19	1
	Tent - 8 person	2017-12-14	1
	Tent - 4 person	2017-10-15	1

HAVING clause

The HAVING clause acts like a WHERE clause but it identifies groups meeting a criterion, rather than rows. A HAVING clause usually follows a GROUP BY clause

- 6) List the department names with more than 2 employees

```
SELECT department.name, COUNT(employee.employeeID)
FROM department
INNER JOIN employee
ON department.DepartmentID = employee.DepartmentID
GROUP BY department.name
HAVING COUNT(employee.employeeid) > 2;
```

Your result set should look like this:

	name	count(employee.employeeid)
▶	Marketing	3

Remember that WHERE works on rows, HAVING works on aggregates (e.g. COUNT, AVG, SUM etc.)

*Better approach (more user friendly and saving CPU resources) is to use **an alias***

```
SELECT department.name,
       COUNT(employee.employeeID) AS NumEmployees
FROM department
INNER JOIN employee
ON department.DepartmentID = employee.DepartmentID
GROUP BY department.name
HAVING NumEmployees > 2;
```

- 7) **TASK.** Find the item id's sold by at least two departments on the second floor

Your result set should look like this:

	ItemID
	14

Remember: If necessary review the slides from the week 5 lecture on the types of joins.

- 8) **TASK.** Type the query to list the departments that have an average salary over \$55000

Your result set should look like this:

	DepartmentID	AverageSalary
	1	125.000.00
	8	60.000.00
	9	79.500.00
	10	75.000.00
	11	64.000.00

Remember that HAVING is the way to use a condition for any column that has an aggregate function used on it (e.g. AVG MAX SUM COUNT etc)

- 9) **TASK.** Type the name of items which have only been delivered by exactly one supplier

HINT: You will need to join three tables (Item, deliveryitem, delivery) together and make sure the ambiguous columns in your select statement are fully qualified, e.g.:

SELECT employee.Lastname, department.Name

FROM employee

INNER JOIN department

ON ...

Your result set should look like this:

Name	SupplierCount
BBO - Jumbuk	1
Boots - Mens Hiking	1
Boots - Womens Goretex	1
Boots - Womens Hiking	1
Boots Riding	1
Camping chair	1
Cowboy Hat	1
Horse saddle	1
Polar Fleece Beanie	1
Sun Hat	1
Tent - 2 person	1
Tent - 4 person	1
Tent - 8 person	1

- 10) **TASK.** List the suppliers that have delivered at least 10 distinct items. List the supplier name and id

Your result set should look like this:

SupplierID	Name
102	Nepalese Corp.
105	All Points Inc.

- 11) **TASK.** Type the SQL that for each item, gives its type, the departments that sell the item, and the floor location of these departments.

HINT: You will need to join four tables (Item, SaleItem, Sale and Department) together and make sure each ambiguous column in your select statement is fully qualified e.g.:

SELECT Employee.lastname, Department.Name

FROM Employee

NATURAL JOIN Department

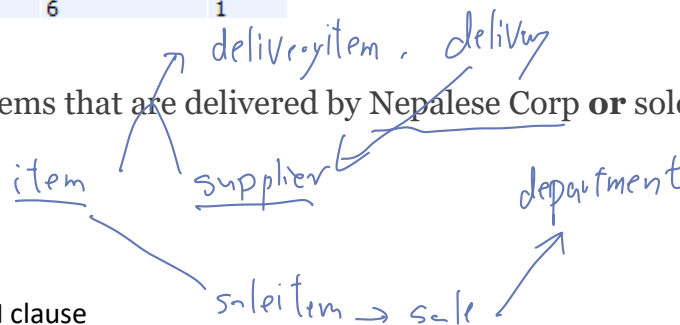
Your result set should look something like this:

Handwritten notes for Task 11:

- item
- name
- type
- saleitem
- sale
- department
- departmentID
- floor

	Name	Type	DepartmentID	Floor
	BBO - Jumbuk	F	4	3
	Boots - Mens Hikina	C	3	2
	Boots - Womens Goretex	C	3	2
	Boots - Womens Hikina	C	3	2
	Boots Ridina	C	2	1
	Camping chair	F	4	3
	Compass - Silva	N	2	1
	Compass - Silva	N	4	3
	Compass - Silva	N	6	1
	Cowboy Hat	C	3	2
	Exploring in 10 Easy Les...	B	2	1
	Geo positioning system	N	2	1
	Geo positioning system	N	6	1
	Gortex Rain Coat	C	2	1
	Gortex Rain Coat	C	3	2
	Gortex Rain Coat	C	4	3
	Gortex Rain Coat	C	5	4
	Gortex Rain Coat	C	6	1
	How to Win Foreign Frie...	B	2	1
	How to Win Foreign Frie...	B	6	1
	Map case	E	6	1
	Map measure	N	6	1
	Pocket knife - Essential	E	2	1
	Pocket knife - Essential	E	3	2
	Pocket knife - Essential	E	4	3
	Pocket knife - Essential	E	5	4
	Pocket knife - Essential	E	6	1
	Pocket knife - Essential	E	7	2
	Polar Fleece Beanie	C	3	2
	Sun Hat	C	3	2
	Tent - 2 person	F	7	2
	Tent - 4 person	F	7	2
	Tent - 8 person	F	7	2
	Torch	E	2	1
	Torch	E	3	2
	Torch	E	4	3
	Torch	E	5	4
	Torch	E	6	1

12) TASK Name the items that are delivered by Nepalese Corp **or** sold in the Navigation department



UNION

You could also use the UNION clause

A UNION Join includes all data from each table that is joined. The columns selected in a UNION join must be the same.

Name
Compass - Silva
Exploring in 10 Easy Lessons
Geo positioning system
How to Win Foreign Friends
Map case
Map measure
Gortex Rain Coat
Pocket knife - Steadfast
Pocket knife - Essential
Camping chair
BBO - Jumbuk
Torch
Tent - 2 person
Tent - 8 person
Tent - 4 person

END OF TUTORIAL 6

Appendix New Department Store Physical ER Model

