

## Tutorial - Week 8 Solutions

### Objectives:

- OUTER JOIN
- UNARY JOINS
- VIEWS
- RELATIONAL DIVIDES

1) Find the number of units sold of each item

```
SELECT item.Name, sum(saleitem.Quantity) as UnitsSold
FROM saleitem NATURAL JOIN item
GROUP BY item.Name
ORDER BY item.Name;
```

Name	UnitsSold
Boots Ridina	4
Compass - Silva	14
Exolorina in 10 Easy Lessons	3
Geo positionina system	7
Sun Hat	10
How to Win Foreign Friends	7
Map case	6
Map measure	10
Gortex Rain Coat	19
Pocket knife - Essential	18
Camping chair	1
BBO - Jumbuk	2
Torch	33
Polar Fleece Beanie	6
Tent - 2 person	5
Tent - 8 person	2
Tent - 4 person	1
Cowboy Hat	1
Boots - Womens Hiking	1
Boots - Womens Goretex	4
Boots - Mens Hiking	2

However, this query **does not** return the fact that the Horse Saddle has not been sold!

### OUTER JOINS

To retrieve all items even if they have not been sold you may need to use an OUTER JOIN.

MySQL Server supports LEFT OUTER JOIN and RIGHT OUTER JOIN. Syntactically while RIGHT JOIN and LEFT JOIN work it is best to use the OUTER word to indicate your intent with the SQL statement.

You use a LEFT OUTER JOIN or a RIGHT OUTER JOIN dependent on where the 'Null' column table resides in your query.

The following query provides a dummy column in the saleitem table.

```
SELECT item.Name, SUM(saleitem.quantity) as UnitsSold
FROM saleitem RIGHT OUTER JOIN item
ON saleitem.itemID = item.itemID
GROUP BY item.name
ORDER BY item.name;
```

Name	UnitsSold
Horse saddle	NULL
Boots Riding	4
Compass - Silva	14
Exploring in 10 Easy Lessons	3
Geo positioning system	7
Sun Hat	10
How to Win Foreign Friends	7
Map case	6
Map measure	10
Gortex Rain Coat	19
Pocket knife - Essential	18
Camping chair	1
BBO - Jumbuk	2
Torch	33
Polar Fleece Beanie	6
Tent - 2 person	5
Tent - 8 person	2
Tent - 4 person	1
Cowboy Hat	1
Boots - Womens Hiking	1
Boots - Womens Goretex	4
Boots - Mens Hiking	2

- 2) Find any suppliers that deliver no more than two unique items. List the suppliers in alphabetical order

```
SELECT supplier.Name, COUNT(DISTINCT deliveryitem.itemID) as
Unique_item_count
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;
```

Name	Unique_Item_Count
Sweatshops Unlimited	2

- 3) Find the names of suppliers that have never delivered a Compass

Name
► Sweatshops Unlimited
Sao Paulo Manufacturing

```
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 `%`  
Therefore it is always good to query reference tables like item to know how the item name is actually stored in the database.

## Unary Joins

The query below is a self join to the employee table. You will notice that we have created an alias for the employee table as emp for employees and boss for their manager. The bossid in the employee table becomes the employeeid in the boss table. This is also known as an *UNARY* join

- 4) List the first names of each manager and their employees. Order the result by manager first name, then employee first name.

```
SELECT boss.FirstName AS Manager, emp.FirstName AS employee
FROM employee AS emp INNER JOIN employee AS boss
ON emp.BossID = boss.employeeID
ORDER BY boss.FirstName, emp.FirstName;
```

Manager	Employee
Alice	Brier
Alice	Ned
Alice	Sophie
Alice	Todd
Andrew	James
Andrew	Mark
Andrew	Pat
Andrew	Paul
Andrew	Saniav
Brier	Sarah
Clare	GiGi
Clare	Maggie
Clare	Rita
Ned	Andrew
Ned	Clare
Todd	Nancy

- 5) TASK Now modify this query to use an outer join to list Alice as an employee

```
SELECT boss.FirstName AS Manager, emp.FirstName AS employee
FROM employee AS emp LEFT OUTER JOIN employee AS boss
ON emp.BossID = boss.employeeID
ORDER BY boss.FirstName, emp.FirstName;
```

	Manager	Employee
▶	<b>NULL</b>	Alice
	Alice	Brier
	Alice	Ned
	Alice	Sophie
	Alice	Todd
	Andrew	James
	Andrew	Mark
	Andrew	Pat
	Andrew	Paul
	Andrew	Sanjay
	Brier	Sarah
	Clare	Gigi
	Clare	Maggie
	Clare	Rita
	Ned	Andrew
	Ned	Clare
	Todd	Nancy

- 6) Type the query to count the number of direct employees of each manager, List the employeeID, Manager Name and number of employees.

Your result set should look similar to this:

	EmployeeID	ENAME	Emp_count
	3	Andrew Jackson	5
	1	Alice Munro	4
	4	Clare Underwood	3
	2	Ned Kelly	2
	5	Todd Beamer	1
	7	Brier Patch	1

```
SELECT boss.employeeID, Concat(boss.FirstName, ' ', boss.LastName) as
ENAME, COUNT(*) Emp_count
FROM employee wrk INNER JOIN employee boss
ON wrk.BossID = boss.employeeID
GROUP BY boss.employeeID, ENAME
ORDER BY EMP_COUNT DESC
```

## Views

Views are a table whose rows are not explicitly stored in the database but are returned as needed from a stored view definition.

Consider the following view

```
CREATE VIEW vdepartment_wages AS
SELECT departmentID, Name, SUM(Salary) as TotalWages
FROM department NATURAL JOIN employee
GROUP BY departmentID, Name
ORDER BY departmentID;
```

This creates a view called vdepartment\_wages. I can use this view like any table in my schema.

```
SELECT *
FROM vdepartment_wages
WHERE TotalWages > 150000;
```

	DepartmentID	Name	TotalWages
	9	Purchasing	159000.00
	11	Marketing	192000.00

However, what is really going on is the following query:

```
SELECT *
FROM
    (SELECT departmentID, Name, SUM(Salary) as TotalWages
     FROM department NATURAL JOIN employee
     GROUP BY departmentID, Name
     ORDER BY departmentID) as vdepartment_Wages
WHERE TotalWages > 150000;
```

The SELECT statement for the view is being used in the FROM clause of SQL. This is here to explain how the view is used by retrieving the stored code from CREATE VIEW statement. This is still considered as a view and is known as an **INLINE VIEW**.

At any time the SQL that makes up the view definition can be queried from the Data Dictionary:

```
SELECT table_name, view_definition
FROM Information_schema.views
-- WHERE Table_SCHEMA= 'labs2018' - remove comment for BYOD devices
;
```

- 7) List the employees in the Accounting department and the difference between their salaries and the average salary of the department

First create a view of the all department Names and average Salary called VdepartmentSalary

```
CREATE VIEW VdepartmentSalary (departmentID, dptavgsal) AS
SELECT departmentID, AVG(Salary)
FROM employee
GROUP BY departmentID;
```

Now use the view vdepartmentSalary in the query to answer the question

```
SELECT FirstName, LastName, FORMAT((Salary - dptavgsal),2)
AS Salary_DeptAvgSalary
FROM vdepartmentSalary INNER JOIN employee INNER JOIN department
ON vdepartmentSalary.departmentID = employee.departmentID
AND department.departmentID = employee.departmentID
WHERE department.Name = 'Accounting';
```

	FirstName	LastName	Salary_DeptAvgSalary
	Todd	Beamer	8.000.00
	Nancy	Cartwright	-8.000.00

- 8) List each employee's salary, the average salary within that person's department, and the difference between the employees' salaries and the average salary of the department  
*HINT: Use the vdepartmentSalary view ...*

```

SELECT FirstName, LastName, Salary, FORMAT(dpavgsal,2) AS DeptAvSal,
       FORMAT(Salary - dpavgsal,2) AS DiffEAvgDSal
FROM vdepartmentSalary NATURAL JOIN employee
WHERE vdepartmentSalary.departmentID = employee.departmentID;

```

	FirstName	LastName	Salary	DeptAvSal	DiffEAvgDSal
	Alice	Munro	125000.00	125.000.00	0.00
	Ned	Kelly	85000.00	64.000.00	21.000.00
	Andrew	Jackson	55000.00	64.000.00	-9.000.00
	Clare	Underwood	52000.00	64.000.00	-12.000.00
	Todd	Beamer	68000.00	60.000.00	8.000.00
	Nancy	Cartwright	52000.00	60.000.00	-8.000.00
	Brier	Patch	73000.00	79.500.00	-6.500.00
	Sarah	Ferrousion	86000.00	79.500.00	6.500.00
	Sophie	Monk	75000.00	75.000.00	0.00
	Sanjay	Patel	45000.00	45.000.00	0.00
	Rita	Skeeter	45000.00	45.000.00	0.00
	Gail	Montez	46000.00	46.000.00	0.00
	Maggie	Smith	46000.00	46.000.00	0.00
	Paul	Innit	41000.00	43.000.00	-2.000.00
	James	Mason	45000.00	43.000.00	2.000.00
	Pat	Clarkson	45000.00	45.000.00	0.00
	Mark	Zhang	45000.00	45.000.00	0.00

- 9) How many supplier – department pairs exist in which the supplier delivers at least one item of type E to the department?

First the view:

```

CREATE VIEW vSupplierdepartment AS
(SELECT DISTINCT SupplierID, departmentID
 FROM delivery NATURAL JOIN deliveryitem NATURAL JOIN item
 WHERE item.Type = 'E' );

```

And now just count the rows in the view:

```

SELECT count(*)
FROM vSupplierdepartment;

```

	count(*)
	17

## Using Views

- 10) Create a VIEW of department names and total number of sales for each department.

```

CREATE VIEW vdepartmentsales AS
SELECT departmentid, departmentname, COUNT(*) as numsales
FROM department NATURAL JOIN sale
GROUP BY departmentid;

```

- 11) Use the view created in Task 10 to identify department names with more than 5 sales. List the department and number of sales.

And to use the view:

```
SELECT *
FROM vdepartmentsales
WHERE numsales > 5;
```

12) Create a view to list the department id, department name, maximum salary, average salary, minimum salary, total salary and number of staff in each department.

```
CREATE VIEW VemployeeSalary AS
SELECT department.departmentID, departmentName, MAX(employeeSalary)
as Maxsal, AVG(employeeSalary) as Avgsal,
MIN(employeeSalary) as Minsal, SUM(employeeSalary) as Totalsal,
COUNT(employee.employeeID) as Headcount
FROM department NATURAL JOIN employee
GROUP BY department.departmentID, departmentName;
```

13) Use the view created in Task 12 to find the lowest salary in the department with the highest headcount.

```
SELECT departmentID, departmentName, minsal
FROM Vemployeesalary
WHERE Headcount in
(SELECT max(Headcount)
FROM Vemployeesalary);
```

You will notice for clarity each view name uses a "v" prefix to indicate it is a view. This is good practice in writing SQL but is not mandatory.

## Relational Divides

### Relational Divides - How they work

14) List the departments that have at least one sale of all the items delivered to them  
Attempt 1 uses NOT EXISTS to find the departments that have sold all itemids that have been delivered.

```
SELECT DISTINCT departmentID
FROM deliveryitem del1
WHERE NOT EXISTS
(SELECT *
FROM deliveryitem del2
WHERE del2.departmentID = del1.departmentID
AND NOT EXISTS
(SELECT *
FROM saleitem NATURAL JOIN sale
WHERE del2.itemID = saleitem.itemID
AND del1.departmentID = sale.departmentID));
```

Firstly NOT EXISTS means if there are no rows in the result set that evaluates to TRUE, if there are rows it evaluates to FALSE. Therefore if the departmentid from deliveryitem del1 matches a row in deliveryitem del2 a value is in the set and there not exists evaluates to FALSE.

It helps to look at the result pairs side by side. First the deliveryitem deapartmentids and itemids:

```
SELECT distinct(departmentid), itemid
FROM deliveryitem
ORDER BY departmentid, itemid;
```

The result set is (departmentid, itemid)

```
{(2,3), (2,5), (2,6), (2,9), (2,12), (2,14), (2,17),
(3,1), (3,8), (3,12), (3,14),(3,17), (3,18), (3,22),(3,23),(3,24),(3,25),
(4,2), (4,3), (4,12), (4,14), (4,15), (4,16), (4,17),
(5,12), (5,14), (5,17),
(6,3), (6,5), (6,6), (6,9), (6,10), (6,11), (6,12), (6,13), (6,14), (6,17),
(7,5), (7,9), (7,14), (7,19), (7,19), (7,20), (7,21) }
```

This automatically tells us that departmentids 1,8,9,10 & 11 will not be in our result set because they have not received a delivery

We then look at the departments that have sold items:

```
SELECT distinct(departmentid), saleitem.itemid
FROM saleitem INNER JOIN sale
on sale.saleid = saleitem.saleid
ORDER BY departmentid, itemid;
```

This result set is (departmentid, itemid):

```
{(2,1), (2,3), (2,5), (2,6), (2,9), (2,12), (2,14), (2,17),
(3,8), (3,12), (3,14), (3,18),(3,22), (3,23), (3,24), (3,25),
(4,3), (4,12), (4,14), (4,15), (4,16), (4,17),
(5, 12), (5,14), (5,17) ,
(6,3), (6,6), (6,9), (6,10), (6,11), (6,12), (6, 14), (6,17),
(7,14), (7,19), (7,20), (7,21)}
```

Consider the result sets side by side - each row is the set for the departmentid, itemid in deliveryitem and sale/saleitem tables:

deliveryitem (departmentid, itemid)		sale/saleitem (departmentid, itemid)
(2,3), (2,5), (2,6), (2,9), (2,12), (2,14), (2,17)		(2,1), <b>(2,3)</b> , <b>(2,5)</b> , <b>(2,6)</b> , <b>(2,9)</b> , <b>(2,12)</b> , <b>(2,14)</b> , <b>(2,17)</b> ,
<b>(3,1)</b> , (3,8), (3,12), (3,14),(3,17), (3,18), (3,22),(3,23),(3,24),(3,25)		(3,8), (3,12), (3,14), (3,18),(3,22), (3,23), (3,24), (3,25),
<b>(4,2)</b> , (4,3), (4,12), (4,14), (4,15), (4,16), (4,17)		(4,3), (4,12), (4,14), (4,15), (4,16), (4,17),
(5,12), (5,14), (5,17)		<b>(5, 12)</b> , <b>(5,14)</b> , <b>(5,17)</b>
(6,3), (6,5), (6,6), (6,9), (6,10), (6,11), (6,12), <b>(6,13)</b> , (6,14), (6,17)		(6,3), (6,6), (6,9), (6,10), (6,11), (6,12), (6, 14), (6,17)
<b>(7,5)</b> , <b>(7,9)</b> , (7,14), (7,19), (7,19), (7,20), (7,21)		(7,14), (7,19), (7,20), (7,21)

Table 1: The result set in deliveryitem must be found for the department result set for sale/saleitem. This is true for departments 2 & 5 only (note the deliveryitem itemID is a subset of the sale/saleitem result set for department id 2, as item id 1 was in stock and sold but has not been delivered)

Consider the department 3 (row 3) result sets.

The SELECT clause is selecting department 3 from the deliveryitem (del1) table it then joins to the deliveryitem (del2) in the first subquery and finds departmentID 3, itemID 1 the result set (3,1). As the record is found the NOT EXISTS condition is evaluated to FALSE as a record exists.

Now we need to find a FALSE record for the sale, however result set (3,1) does NOT EXIST in the sale, saleitem subquery - and evaluates to TRUE. Because it is an AND condition both subqueries must be true TRUE != FALSE the result set is not returned.



This process repeats for every result set returned by the queries. Only when FALSE = FALSE (rows DO EXIST) will a result set be returned. This is because of the join to the table deliveryitem (aliased as del1) in both subqueries del1.departmentid=del2.departmentid in subquery 1 and del1.departmentid=sale.departmentid in subquery 2.

15) Find the items (itemID) sold by ALL departments located on the second floor

```
SELECT saleitem.itemID
FROM saleitem NATURAL JOIN sale NATURAL JOIN department
WHERE department.Floor = 2
GROUP BY saleitem.itemID
HAVING count(DISTINCT department.departmentID) =
        (SELECT count(DISTINCT departmentID)
         FROM department
         WHERE department.Floor = 2
        )
ORDER BY saleitem.itemID;
```

And using a different method

```
SELECT DISTINCT itemID
FROM item
WHERE NOT EXISTS
    (SELECT *
     FROM department
     WHERE department.Floor = 2
     AND NOT EXISTS
        (SELECT *
         FROM saleitem NATURAL JOIN sale
         WHERE saleitem.itemID = item.itemID
         AND sale.departmentID = department.departmentID
        )
    )
ORDER BY itemID;
```

ItemID
14
NULL

16) List the department names that have not recorded a sale for all the items of type N

```
SELECT department.Name
FROM department
WHERE departmentID NOT IN
    (SELECT departmentID
     FROM department
     WHERE NOT EXISTS
        (SELECT *
         FROM item
         WHERE item.Type = 'N'
         AND NOT EXISTS
            (SELECT *
             FROM sale NATURAL JOIN saleitem
            )
        )
    )
```

```

WHERE sale.departmentID =
department.departmentID
AND saleitem.itemID = item.itemID)
)
ORDER BY department.Name;

```

Name
Accounting
Books
Clothes
Equipment
Furniture
Management
Marketing
Personnel
Purchasing
Recreation

17) Type a relational divide query that lists the suppliers that delivery only items sold by the Books department

Name
Sweatshoos Unlimited
Sao Paulo Manufacturing

```

SELECT supplier.Name
FROM supplier
WHERE SupplierID IN
  (SELECT SupplierID
   FROM delivery)
AND NOT EXISTS
  (SELECT *
   FROM deliveryitem NATURAL JOIN delivery
   WHERE delivery.SupplierID = supplier.SupplierID
   AND itemID NOT IN
    (SELECT itemID
     FROM saleitem NATURAL JOIN sale NATURAL JOIN department
     WHERE department.Name = 'Books'));

```

As you will see there are many different queries that can achieve the same result set.

**End of Week 9 Lab**

## Appendix: The New department Store ER Physical Model

