INFO90002 Week 7 Tutorial

Objectives:

- Learn SQL by Practice
- OUTER JOIN syntax
- UNARY JOINS
- VIEWS
- RELATIONAL DIVIDES

Section 1 SQL

Connect to your MySQL database on the engineering server

1.1 Find the number of units sold of each item

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

Name	UnitsSold
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
BBQ - 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 saleitem.ItemID
ORDER BY saleitem.ItemID;
```

	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
	BBQ - 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

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

Your result set should look something like this:

	Name	Unique_Item_Count
	Sweatshops Unlimited	2

1.3 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 %

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

1.4 Find, for each department, its floor and the average salary in the department.

Your result set should look something like this:

	Name	Floor	AVG_SAL
	Accounting	5	68.000.00
	Books	1	45.000.00
	Clothes	2	46.000.00
	Equipment	3	43.000.00
	Furniture	4	45.000.00
	Management	5	125.000.00
	Marketing	5	64.000.00
	Navigation	1	45.000.00
	Personnel	5	75.000.00
	Purchasing	5	70.333.33
	Recreation	2	45.000.00

1.5 Type the query to find the Items (ItemID) sold on floors other than the second floor

The result set should look similar to this:

	ItemID
	1
	3
	5
	6
	9
	10
	11
	15
	16

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.

1.6 List the names of each manager and their employees arranged by manager's name and employee's name within manager.

```
SELECT boss.FirstName AS Manager, emp.FirstName AS Employee
  FROM employee AS emp, employee AS boss
  WHERE 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

1.7 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
NULL	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

1.8 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

1.9 Find the department/s that sell at least 4 different items

Your result set should look similar to this:

Name
Books
Clothes
Equipment
Navigation
Recreation

1.10 Find the departments that sell at least 4 items and list how many items each department sells (Alternative wording: Find the departments that have made 4 or more transactions and list how many transactions each department has made.)

```
SELECT department.Name, count(saleitem.itemid) as
Sale_Quantity
FROM department INNER JOIN sale INNER JOIN saleitem
ON department.DepartmentID = sale.DepartmentID AND
```

```
sale.SaleID = saleitem.SaleID
GROUP BY department.Name
HAVING COUNT(DISTINCT(saleitem.itemid)) >= 4;
```

Name	Sale_Quantity
Books	23
Clothes	27
Equipment	17
Navigation	39
Recreation	10

1.11 Type the query to find the Items (ItemID) sold on floors other than the second floor

The result set should look similar to this:

ItemID
1
3
5
6
9
10
11
15
16

1.12 Find the departments that have never sold a geo positioning system

The result set should look similar to this:

Name	departmentid
Books	2
Clothes	3
Equipment	4
Furniture	5
Navigation	6
Recreation	7

1.13 List the department id and average salary where the average salary of the employees of each manager is more than $55000

The result set should look similar to this:

DepartmentID	AvgSalary
9	86.000.00

1.14 Find the supplier id and supplier names that do not deliver compasses

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

SupplierID	Name
104	Sweatshops Unlimited
106	Sao Paulo Manufacturing
NULL	NULL

1.15 Type the query to find the total quantity sold of each item by the departments on the second floor

The result set should look similar to this:

Name	TOTAL_SALES
Sun Hat	10
Pocket knife - Essential	9
Torch	8
Polar Fleece Beanie	6
Tent - 2 person	5
Boots - Womens Goretex	4
Tent - 8 person	2
Gortex Rain Coat	2
Boots - Mens Hiking	2
Boots - Womens Hiking	1
Tent - 4 person	1
Cowboy Hat	1

Section 2: Views

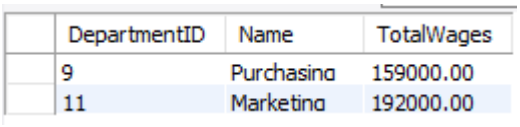
Views are a table whose rows are not explicitly stored in the database but are returned as needed from a stored view definition.

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

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

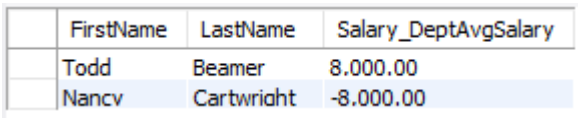
```
SELECT table_name, view_definition
FROM Information_schema.views
-- WHERE Table_SCHEMA= 'labs2018' - remove comment for BYOD
devices
;
```

2.1 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 Department Name and average Salary

And use the view in the query to answer the question

The result set should look similar to this:



	FirstName	LastName	Salary_DeptAvgSalary
	Todd	Beamer	8,000.00
	Nancy	Cartwright	-8,000.00

2.2 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: You can reuse the view created in 2.1

The result set should look similar to this:

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	Fergusson	86000.00	79.500.00	6.500.00
Sophie	Monk	75000.00	75.000.00	0.00
Saniav	Patel	45000.00	45.000.00	0.00
Rita	Skeeter	45000.00	45.000.00	0.00
Gigi	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

2.3 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

2.4 Which department (departmentid) has more than five sales? Use views in your answer

2.5 Create a view to list the maximum salary, average salary, minimum salary, total salary and number of staff in each department. Find the lowest salary in the department with the highest headcount.

Section 3 Relational Divides

Relational Divides - How they work

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 deliveritem del1 matches a row in deliveritem 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 (departmnetid, itemid)		Sale/SaleItem (departmentid, itemid)
(2,3), (2,5), (2,6), (2,9), (2,12), (2,14), (2,17)		(2,1), (**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)		(3,8), (3,12), (3,14), (3,18),(3,22), (3,23), (3,24), (3,25),
(**4,2**), (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)		(**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)		(6,3), (6,6), (6,9), (6,10), (6,11), (6,12), (6, 14), (6,17)
(**7,5**), (**7,9**), (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.

3.1 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

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

The result set should look similar to this:

	Name
	Accounting
	Books
	Clothes
	Equipment
	Furniture
	Management
	Marketing
	Personnel
	Purchasing
	Recreation

3.3 Who are the supplier id and supplier names that deliver all the items of type N?

The result set should look similar to this:

	SupplierID	Name
	101	Global Books & Maps
	102	Nepalese Corp.
	103	All Sports Manufacturing
	NULL	NULL

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

	Name
	Sweatshops Unlimited
	Sao Paulo Manufacturing

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

End of Week 7 Lab

Appendix: The New Department Store ER Physical Model