# Database Labs Case Study

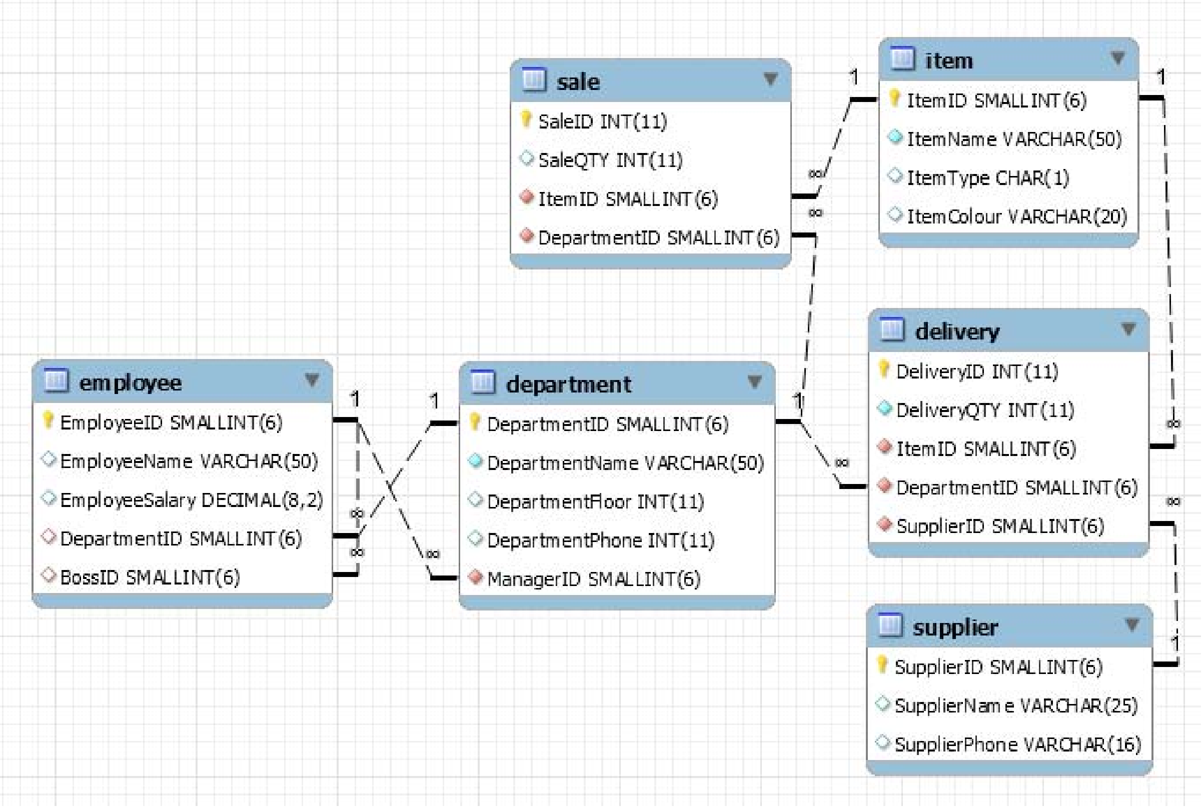
This document contains the details of the database that you will set up to do your lab work.
It contains four sections:

There are around **16 SQL** queries for you to complete each week.

Before you can run these queries in this document you must download from LMS and run in MySQL Workbench the script to create the tables and populate them with data.

Answers for the SQL writing questions will be provided around week 8.

## Section 1: Physical data model



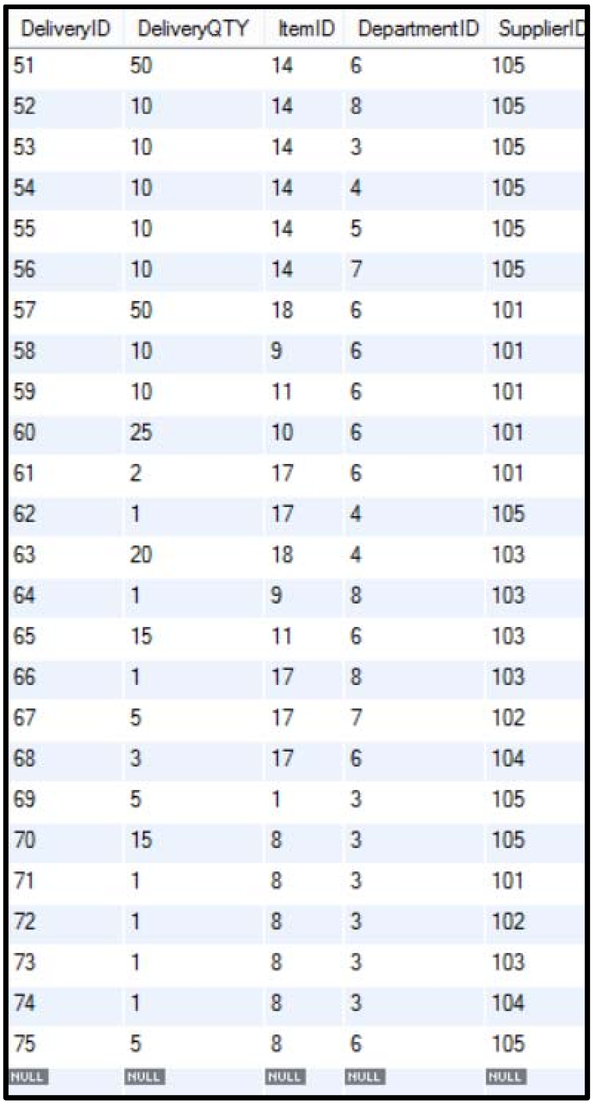
This database is the core of an information system used to manage a large city department store.

The store has several departments, both retail and administrative, located on different floors of a building. Each department has several employees working for it, and one boss who manages it: details of employees and bosses are stored in the same table.

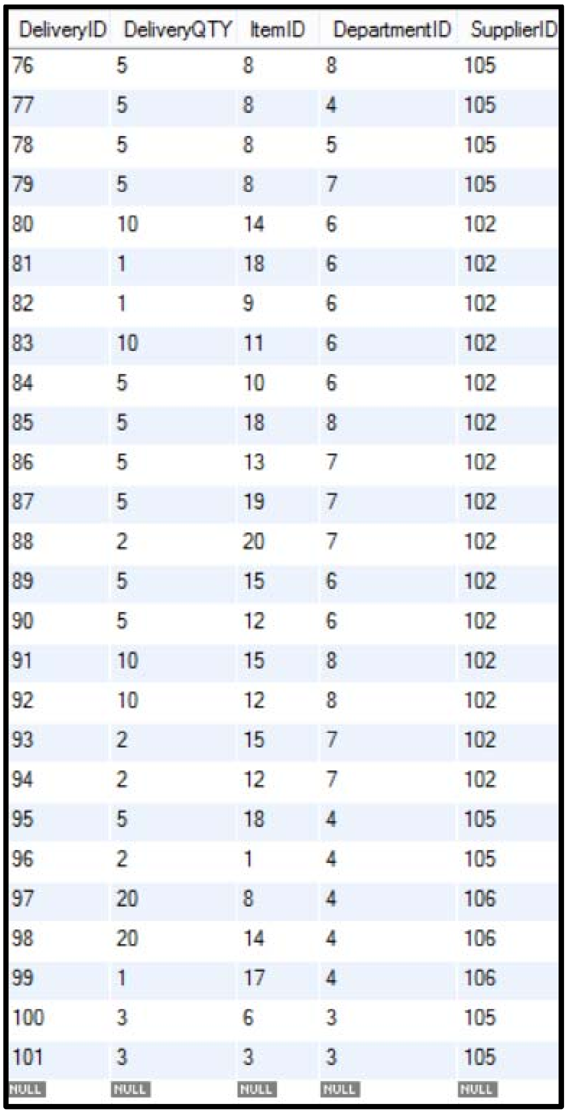
The retail departments make sales. The items they sell are delivered by suppliers.
Details of sales, suppliers and deliveries are recorded in the database.

# Section 2: Data in the tables

### Delivery

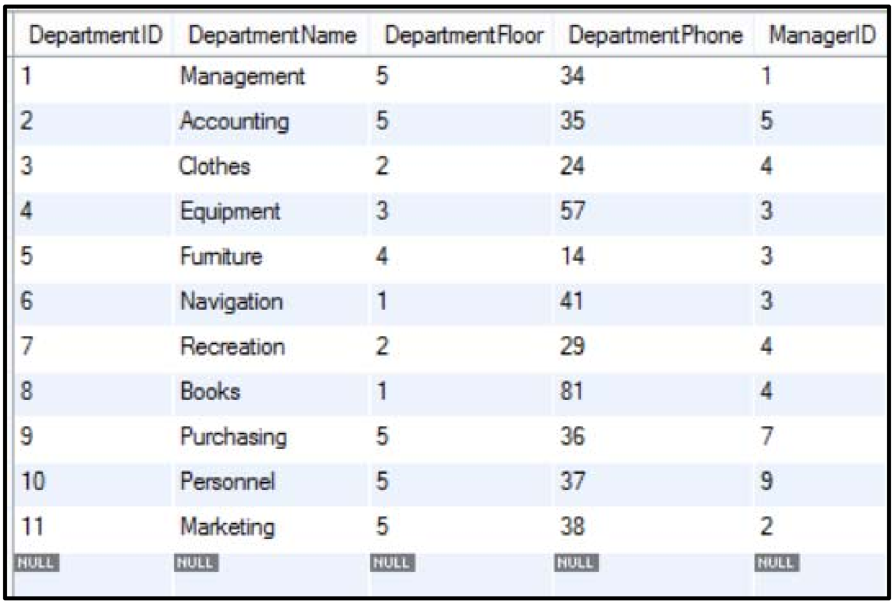


| DeliveryID | DeliveryQTY | ItemID | DepartmentID | SupplierID |
|---|---|---|---|---|
| 51 | 50 | 14 | 6 | 105 |
| 52 | 10 | 14 | 8 | 105 |
| 53 | 10 | 14 | 3 | 105 |
| 54 | 10 | 14 | 4 | 105 |
| 55 | 10 | 14 | 5 | 105 |
| 56 | 10 | 14 | 7 | 105 |
| 57 | 50 | 18 | 6 | 101 |
| 58 | 10 | 9 | 6 | 101 |
| 59 | 10 | 11 | 6 | 101 |
| 60 | 25 | 10 | 6 | 101 |
| 61 | 2 | 17 | 6 | 101 |
| 62 | 1 | 17 | 4 | 105 |
| 63 | 20 | 18 | 4 | 103 |
| 64 | 1 | 9 | 8 | 103 |
| 65 | 15 | 11 | 6 | 103 |
| 66 | 1 | 17 | 8 | 103 |
| 67 | 5 | 17 | 7 | 102 |
| 68 | 3 | 17 | 6 | 104 |
| 69 | 5 | 1 | 3 | 105 |
| 70 | 15 | 8 | 3 | 105 |
| 71 | 1 | 8 | 3 | 101 |
| 72 | 1 | 8 | 3 | 102 |
| 73 | 1 | 8 | 3 | 103 |
| 74 | 1 | 8 | 3 | 104 |
| 75 | 5 | 8 | 6 | 105 |
| NULL | NULL | NULL | NULL | NULL |



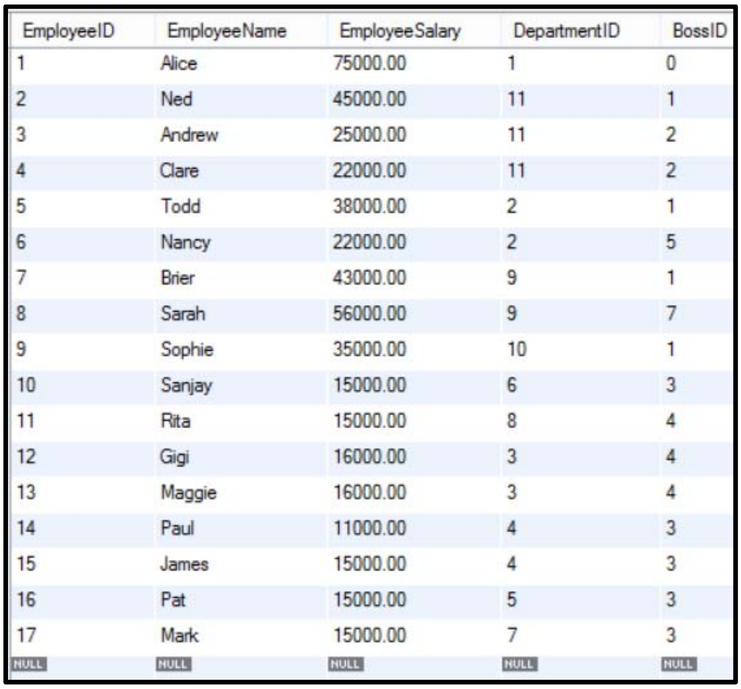
| DeliveryID | DeliveryQTY | ItemID | DepartmentID | SupplierID |
|---|---|---|---|---|
| 76 | 5 | 8 | 8 | 105 |
| 77 | 5 | 8 | 4 | 105 |
| 78 | 5 | 8 | 5 | 105 |
| 79 | 5 | 8 | 7 | 105 |
| 80 | 10 | 14 | 6 | 102 |
| 81 | 1 | 18 | 6 | 102 |
| 82 | 1 | 9 | 6 | 102 |
| 83 | 10 | 11 | 6 | 102 |
| 84 | 5 | 10 | 6 | 102 |
| 85 | 5 | 18 | 8 | 102 |
| 86 | 5 | 13 | 7 | 102 |
| 87 | 5 | 19 | 7 | 102 |
| 88 | 2 | 20 | 7 | 102 |
| 89 | 5 | 15 | 6 | 102 |
| 90 | 5 | 12 | 6 | 102 |
| 91 | 10 | 15 | 8 | 102 |
| 92 | 10 | 12 | 8 | 102 |
| 93 | 2 | 15 | 7 | 102 |
| 94 | 2 | 12 | 7 | 102 |
| 95 | 5 | 18 | 4 | 105 |
| 96 | 2 | 1 | 4 | 105 |
| 97 | 20 | 8 | 4 | 106 |
| 98 | 20 | 14 | 4 | 106 |
| 99 | 1 | 17 | 4 | 106 |
| 100 | 3 | 6 | 3 | 105 |
| 101 | 3 | 3 | 3 | 105 |
| NULL | NULL | NULL | NULL | NULL |

### Department



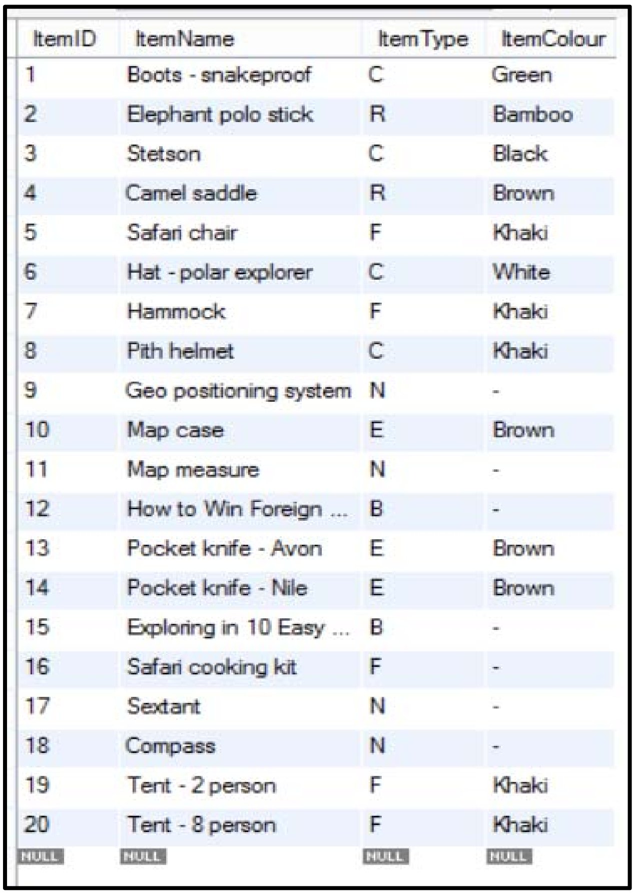
| DepartmentID | DepartmentName | DepartmentFloor | DepartmentPhone | ManagerID |
|---|---|---|---|---|
| 1 | Management | 5 | 34 | 1 |
| 2 | Accounting | 5 | 35 | 5 |
| 3 | Clothes | 2 | 24 | 4 |
| 4 | Equipment | 3 | 57 | 3 |
| 5 | Furniture | 4 | 14 | 3 |
| 6 | Navigation | 1 | 41 | 3 |
| 7 | Recreation | 2 | 29 | 4 |
| 8 | Books | 1 | 81 | 4 |
| 9 | Purchasing | 5 | 36 | 7 |
| 10 | Personnel | 5 | 37 | 9 |
| 11 | Marketing | 5 | 38 | 2 |
| NULL | NULL | NULL | NULL | NULL |

## Employee



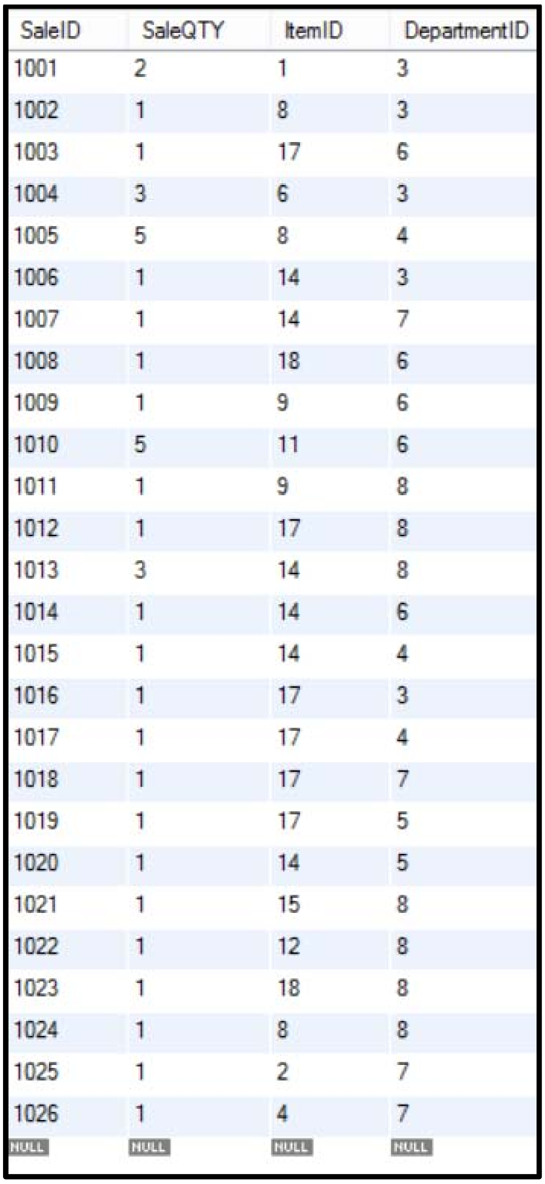
| EmployeeID | EmployeeName | EmployeeSalary | DepartmentID | BossID |
|---|---|---|---|---|
| 1 | Alice | 75000.00 | 1 | 0 |
| 2 | Ned | 45000.00 | 11 | 1 |
| 3 | Andrew | 25000.00 | 11 | 2 |
| 4 | Clare | 22000.00 | 11 | 2 |
| 5 | Todd | 38000.00 | 2 | 1 |
| 6 | Nancy | 22000.00 | 2 | 5 |
| 7 | Brier | 43000.00 | 9 | 1 |
| 8 | Sarah | 56000.00 | 9 | 7 |
| 9 | Sophie | 35000.00 | 10 | 1 |
| 10 | Sanjay | 15000.00 | 6 | 3 |
| 11 | Rita | 15000.00 | 8 | 4 |
| 12 | Gigi | 16000.00 | 3 | 4 |
| 13 | Maggie | 16000.00 | 3 | 4 |
| 14 | Paul | 11000.00 | 4 | 3 |
| 15 | James | 15000.00 | 4 | 3 |
| 16 | Pat | 15000.00 | 5 | 3 |
| 17 | Mark | 15000.00 | 7 | 3 |
| NULL | NULL | NULL | NULL | NULL |

## Item



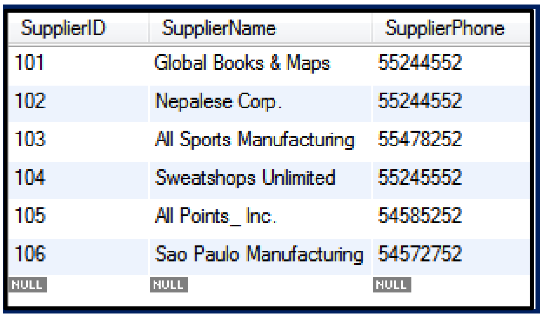
| ItemID | ItemName | ItemType | ItemColour |
|---|---|---|---|
| 1 | Boots - snakeproof | C | Green |
| 2 | Elephant polo stick | R | Bamboo |
| 3 | Stetson | C | Black |
| 4 | Camel saddle | R | Brown |
| 5 | Safari chair | F | Khaki |
| 6 | Hat - polar explorer | C | White |
| 7 | Hammock | F | Khaki |
| 8 | Pith helmet | C | Khaki |
| 9 | Geo positioning system | N | - |
| 10 | Map case | E | Brown |
| 11 | Map measure | N | - |
| 12 | How to Win Foreign ... | B | - |
| 13 | Pocket knife - Avon | E | Brown |
| 14 | Pocket knife - Nile | E | Brown |
| 15 | Exploring in 10 Easy ... | B | - |
| 16 | Safari cooking kit | F | - |
| 17 | Sextant | N | - |
| 18 | Compass | N | - |
| 19 | Tent - 2 person | F | Khaki |
| 20 | Tent - 8 person | F | Khaki |
| NULL | NULL | NULL | NULL |

## Sale



| SaleID | SaleQTY | ItemID | DepartmentID |
|---|---|---|---|
| 1001 | 2 | 1 | 3 |
| 1002 | 1 | 8 | 3 |
| 1003 | 1 | 17 | 6 |
| 1004 | 3 | 6 | 3 |
| 1005 | 5 | 8 | 4 |
| 1006 | 1 | 14 | 3 |
| 1007 | 1 | 14 | 7 |
| 1008 | 1 | 18 | 6 |
| 1009 | 1 | 9 | 6 |
| 1010 | 5 | 11 | 6 |
| 1011 | 1 | 9 | 8 |
| 1012 | 1 | 17 | 8 |
| 1013 | 3 | 14 | 8 |
| 1014 | 1 | 14 | 6 |
| 1015 | 1 | 14 | 4 |
| 1016 | 1 | 17 | 3 |
| 1017 | 1 | 17 | 4 |
| 1018 | 1 | 17 | 7 |
| 1019 | 1 | 17 | 5 |
| 1020 | 1 | 14 | 5 |
| 1021 | 1 | 15 | 8 |
| 1022 | 1 | 12 | 8 |
| 1023 | 1 | 18 | 8 |
| 1024 | 1 | 8 | 8 |
| 1025 | 1 | 2 | 7 |
| 1026 | 1 | 4 | 7 |
| NULL | NULL | NULL | NULL |

## Supplier



| SupplierID | SupplierName | SupplierPhone |
|---|---|---|
| 101 | Global Books & Maps | 55244552 |
| 102 | Nepalese Corp. | 55244552 |
| 103 | All Sports Manufacturing | 55478252 |
| 104 | Sweatshops Unlimited | 55245552 |
| 105 | All Points_ Inc. | 54585252 |
| 106 | Sao Paulo Manufacturing | 54572752 |
| NULL | NULL | NULL |

# Section 3: Example SQL queries

In this section there are about 60 queries that you can analyse and run yourself. Each includes:

- The task
  - these vary in difficulty and are colour coded
    - green – easy
    - blue - moderate
    - red – hard
- Comments
- A sample SQL query that answers the question
- The result of the query

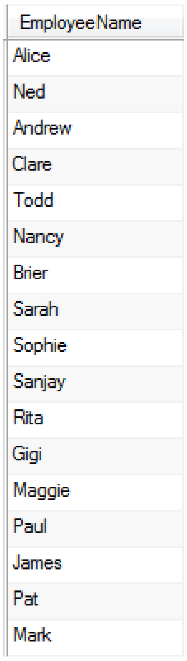
Often there are several possible solutions, which is why we show the result so you can check your query if you do it differently. If a question is challenging, break it down into components.

## Week 3

### 1. Find the names of all employees

This is the most simple query we can do

```
SELECT EmployeeName  FROM Employee;
```

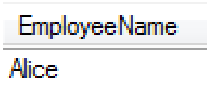


| EmployeeName |
|---|
| Alice |
| Ned |
| Andrew |
| Clare |
| Todd |
| Nancy |
| Brier |
| Sarah |
| Sophie |
| Sanjay |
| Rita |
| Gigi |
| Maggie |
| Paul |
| James |
| Pat |
| Mark |

### 1.1. Find the names of all employees in department number 1

Now add a condition to limit how many rows are returned, and to restrict our result to one column.

```
SELECT EmployeeName FROM Employee
        WHERE DepartmentID = 1;
```

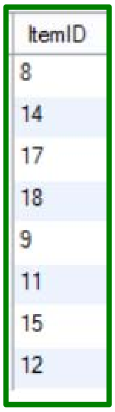


| EmployeeName |
|---|
| Alice |

## 2. Find the items sold on floors other than the second floor

First look at the slightly simpler query: Which items are sold on the second floor?

```
SELECT DISTINCT ItemID
   FROM Sale INNER JOIN Department
        ON Sale.DepartmentID = Department.DepartmentID
                WHERE DepartmentFloor = 2;
```



| ItemID |
|---|
| 8 |
| 14 |
| 17 |
| 18 |
| 9 |
| 11 |
| 15 |
| 12 |

It’s simple now to change this to: Which items are sold on OTHER floors?

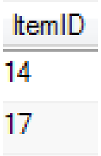
```
SELECT DISTINCT ItemID FROM Sale INNER JOIN Department
        ON Sale.DepartmentID = Department.DepartmentID
                WHERE DepartmentFloor <> 2;
```

The Natural Join syntax is simpler:

```
SELECT DISTINCT ItemID
FROM Sale NATURAL JOIN Department
        WHERE DepartmentFloor <> 2;
```

Variation: find the items sold by at least two departments on the second floor

```
SELECT ItemID FROM Sale NATURAL JOIN Department
        WHERE DepartmentFloor = 2
        GROUP BY ItemID
                HAVING COUNT(DISTINCT Department.DepartmentID) > 1;
```

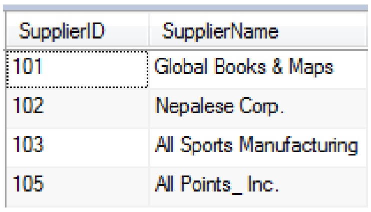


| ItemID |
|---|
| 14 |
| 17 |

## 3. Find the suppliers that deliver compasses

Join the tables, look for compasses and tell us who the suppliers are.

```
SELECT DISTINCT Supplier.SupplierID, SupplierName
FROM Supplier INNER JOIN Delivery INNER JOIN Item
      ON Supplier.SupplierID = Delivery.SupplierID
      AND Item.ItemID = Delivery.ItemID
   WHERE ItemName = 'Compass';
```

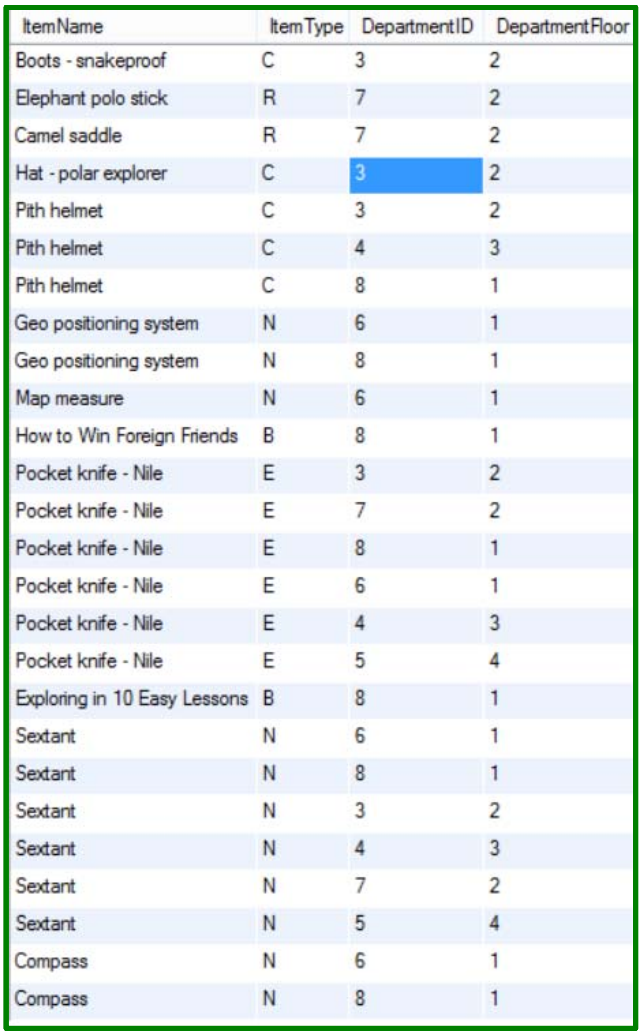


| SupplierID | SupplierName |
|---|---|
| 101 | Global Books & Maps |
| 102 | Nepalese Corp. |
| 103 | All Sports Manufacturing |
| 105 | All Points_ Inc. |

## 4. For each item, give its type, the departments that sell the item, and the floor location of these departments

A simple three way join

```
SELECT Item.ItemName, ItemType,
       Department.DepartmentID, DepartmentFloor
   FROM Item INNER JOIN Sale INNER JOIN Department
   ON Sale.ItemID = Item.ItemID
   AND Sale.DepartmentID =
          Department.DepartmentID;
```

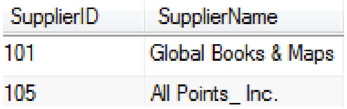


| ItemName | ItemType | DepartmentID | DepartmentFloor |
|---|---|---|---|
| Boots - snakeproof | C | 3 | 2 |
| Elephant polo stick | R | 7 | 2 |
| Camel saddle | R | 7 | 2 |
| Hat - polar explorer | C | 3 | 2 |
| Pith helmet | C | 3 | 2 |
| Pith helmet | C | 4 | 3 |
| Pith helmet | C | 8 | 1 |
| Geo positioning system | N | 6 | 1 |
| Geo positioning system | N | 8 | 1 |
| Map measure | N | 6 | 1 |
| How to Win Foreign Friends | B | 8 | 1 |
| Pocket knife - Nile | E | 3 | 2 |
| Pocket knife - Nile | E | 7 | 2 |
| Pocket knife - Nile | E | 8 | 1 |
| Pocket knife - Nile | E | 6 | 1 |
| Pocket knife - Nile | E | 4 | 3 |
| Pocket knife - Nile | E | 5 | 4 |
| Exploring in 10 Easy Lessons | B | 8 | 1 |
| Sextant | N | 6 | 1 |
| Sextant | N | 8 | 1 |
| Sextant | N | 3 | 2 |
| Sextant | N | 4 | 3 |
| Sextant | N | 7 | 2 |
| Sextant | N | 5 | 4 |
| Compass | N | 6 | 1 |
| Compass | N | 8 | 1 |

## 5. List suppliers that deliver a total quantity of items of types C and N that is altogether greater than 40

A simple query again – but with a compound condition for items C and N

```
SELECT Delivery.SupplierID, SupplierName FROM Supplier
       INNER JOIN Delivery INNER JOIN Item
   ON Supplier.SupplierID = Delivery.SupplierID
   AND Item.ItemID = Delivery.ItemID
   WHERE (ItemType = 'C' OR ItemType = 'N')
   GROUP BY Delivery.SupplierID, SupplierName
       HAVING SUM(DeliveryQTY) > 40;
```

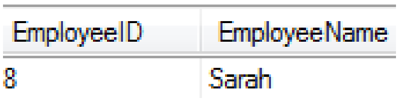


| SupplierID | SupplierName |
|---|---|
| 101 | Global Books & Maps |
| 105 | All Points_ Inc. |

## 6. Find numbers and names of those employees who make more than their manager

Make two copies of the same table (here called *emp* and *boss*) and join them (just like any other join)

```
SELECT emp.EmployeeID, emp.EmployeeName
        FROM Employee emp INNER JOIN Employee boss
        ON emp.BossID = boss.EmployeeID
        WHERE boss.EmployeeSalary < emp.EmployeeSalary;
```

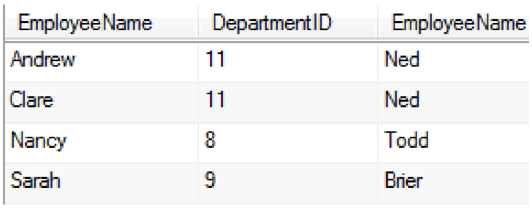


| EmployeeID | EmployeeName |
|---|---|
| 8 | Sarah |

## 7. Find the names of employees who are in the same department as their manager (as an employee). Report the name of the employee, the department, and the boss's name

Another join, but on more than one column.

```
SELECT wrk.EmployeeName, wrk.DepartmentID,
        boss.EmployeeName
        FROM Employee wrk INNER JOIN Employee boss
        ON wrk.BossID = boss.EmployeeID
        WHERE wrk.DepartmentID = boss.DepartmentID;
```



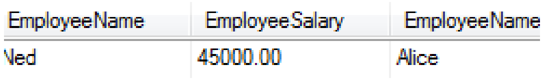
| EmployeeName | DepartmentID | EmployeeName |
|---|---|---|
| Andrew | 11 | Ned |
| Clare | 11 | Ned |
| Nancy | 8 | Todd |
| Sarah | 9 | Brier |

## 8. List the name, salary, and manager of the employees of department 11 who have a salary over $25,000

This is a self join – Join the bosses and workers into a table, and then restrict which rows are returned.

Look at the output of **SELECT * FROM Employee wrk, Employee boss WHERE wrk.BossID = boss.EmployeeID** to see the single table before we remove rows.

```
SELECT wrk.EmployeeName, wrk.EmployeeSalary,
        boss.EmployeeName
    FROM Employee wrk INNER JOIN Employee boss
    ON wrk.BossID = boss.EmployeeID
        WHERE wrk.DepartmentID = 11
        AND wrk.EmployeeSalary > 25000;
```



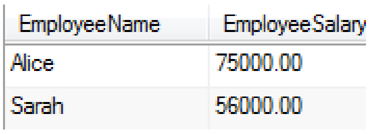
| EmployeeName | EmployeeSalary | EmployeeName |
|---|---|---|
| Ned | 45000.00 | Alice |

# Week 4

## 9. List the names of the employees who earn more than any employee in the marketing department

Split the problem up into 2 components – calculate max salary of people in marketing (inner query), and then find who earns more than that (outer query).

```
SELECT EmployeeName, EmployeeSalary FROM Employee
    WHERE EmployeeSalary >
            (SELECT MAX(EmployeeSalary)
               FROM Employee NATURAL JOIN Department
               WHERE DepartmentName = "Marketing ");
```

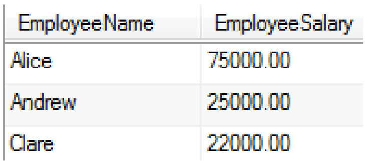


| EmployeeName | EmployeeSalary |
|---|---|
| Alice | 75000.00 |
| Sarah | 56000.00 |

## 10. List the name and salary of the managers with more than two employees

First write code for to find managers with > 2 employees (the sub query).
Then wrap the extra code around it to get the details of the managers

```
SELECT EmployeeName, EmployeeSalary FROM Employee
WHERE EmployeeID IN
    (SELECT BossID FROM Employee
            GROUP BY BossID
                    HAVING COUNT(*) > 2);
```

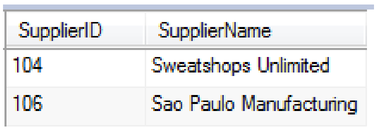


| EmployeeName | EmployeeSalary |
|---|---|
| Alice | 75000.00 |
| Andrew | 25000.00 |
| Clare | 22000.00 |

## 11. Find the suppliers that do not deliver compasses

Work out which suppliers supply compasses (the inner query) and then list the suppliers not in that list (outer query).

```
SELECT SupplierID, SupplierName FROM Supplier
   WHERE SupplierID NOT IN
      (SELECT SupplierID FROM Delivery
            NATURAL JOIN Item
         WHERE ItemName = 'Compass');
```

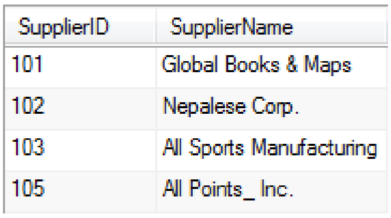


| SupplierID | SupplierName |
|---|---|
| 104 | Sweatshops Unlimited |
| 106 | Sao Paulo Manufacturing |

## 12. List the suppliers that deliver at least two items of type N to departments

Use this one for any query that asks for at least some number…Use DISTINCT to handle multiple deliveries of the same item.

```
SELECT Supplier.SupplierID, SupplierName FROM Delivery
            NATURAL JOIN Supplier NATURAL JOIN Item
    WHERE ItemType = 'N'
    GROUP BY Supplier.SupplierID, SupplierName
    HAVING COUNT(DISTINCT Delivery.ItemID) > 1;
```



| SupplierID | SupplierName |
|---|---|
| 101 | Global Books & Maps |
| 102 | Nepalese Corp. |
| 103 | All Sports Manufacturing |
| 105 | All Points_ Inc. |

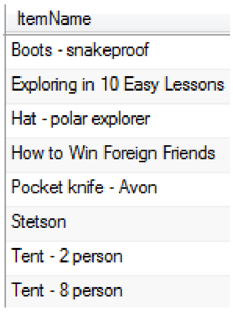
or, including the number we are interested in:

```
SELECT Supplier.SupplierID, SupplierName, count(Distinct delivery.itemid) as NumItems
    FROM Delivery NATURAL JOIN Supplier NATURAL JOIN Item
    WHERE ItemType = 'N'
    GROUP BY Supplier.SupplierID, SupplierName
    HAVING NumItems > 1;
```

## 13. List the items delivered by exactly one supplier
## (i.e. items that are always delivered by the same supplier)

Use Group By with an exact count.

```
    SELECT ItemName FROM Delivery NATURAL JOIN Item
        GROUP BY ItemName
        HAVING COUNT(DISTINCT SupplierID) = 1;
```



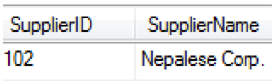
| ItemName |
|---|
| Boots - snakeproof |
| Exploring in 10 Easy Lessons |
| Hat - polar explorer |
| How to Win Foreign Friends |
| Pocket knife - Avon |
| Stetson |
| Tent - 2 person |
| Tent - 8 person |

Can you include the supplier in the output?

## 14. List the suppliers that deliver at least 10 items

Group By with count condition again

```
    SELECT Supplier.SupplierID, SupplierName
        FROM Delivery NATURAL JOIN Supplier
        GROUP BY Supplier.SupplierID, SupplierName
        HAVING COUNT(DISTINCT Delivery.ItemID) >= 10;
```



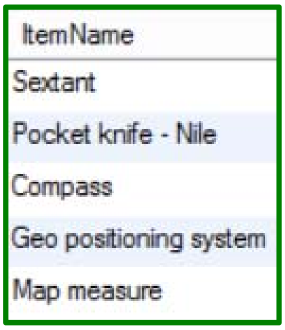
| SupplierID | SupplierName |
|---|---|
| 102 | Nepalese Corp. |

Can you include in the output the number of items they supply?

## 15. List items that are delivered by Nepalese Corp and sold in the Navigation department

Use AND to combine two simple queries together

```
SELECT DISTINCT ItemName FROM Item
    WHERE ItemID IN
        (SELECT ItemID FROM Delivery NATURAL JOIN Supplier
            WHERE SupplierName = 'Nepalese corp.')
    AND ItemID IN
        (SELECT ItemID FROM Sale NATURAL JOIN Department
            WHERE DepartmentName = 'Navigation');
```

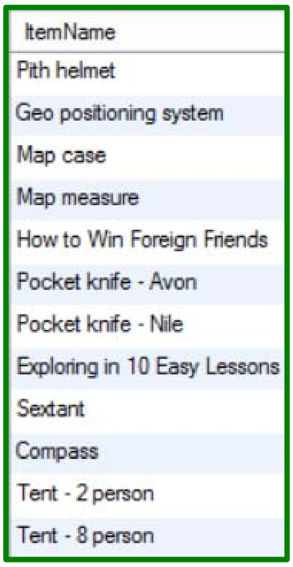


| ItemName |
|---|
| Sextant |
| Pocket knife - Nile |
| Compass |
| Geo positioning system |
| Map measure |

## 16. List items that are delivered by Nepalese Corp *or* sold in the Navigation department

Similar to question 15, but the condition is OR

```
SELECT DISTINCT ItemName FROM Item
    WHERE ItemID IN
        (SELECT ItemID FROM Delivery NATURAL JOIN Supplier
            WHERE SupplierName = 'Nepalese Corp.')
    OR ItemID IN
        (SELECT ItemID FROM Sale NATURAL JOIN Department
            WHERE DepartmentName = 'Navigation');
```

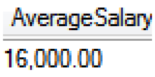


| ItemName |
|---|
| Pith helmet |
| Geo positioning system |
| Map case |
| Map measure |
| How to Win Foreign Friends |
| Pocket knife - Avon |
| Pocket knife - Nile |
| Exploring in 10 Easy Lessons |
| Sextant |
| Compass |
| Tent - 2 person |
| Tent - 8 person |

## 17. Find the average salary of the employees in the Clothes department

A simple join with an Average (and an Alias)

```
SELECT FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
        FROM Employee INNER JOIN Department
            ON Department.DepartmentID = Employee.DepartmentID
        WHERE DepartmentName = 'Clothes';
```

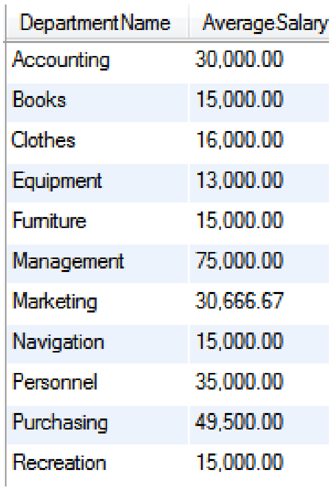


| AverageSalary |
|---|
| 16,000.00 |

### 18. Find, for each department, the average salary of employees

same as 17, except use Group instead of Where

```
SELECT DepartmentName, FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
    FROM Employee INNER JOIN Department
        ON Department.DepartmentID = Employee.DepartmentID
    GROUP BY DepartmentName;
```

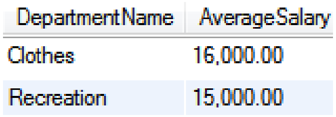


| DepartmentName | AverageSalary |
|---|---|
| Accounting | 30,000.00 |
| Books | 15,000.00 |
| Clothes | 16,000.00 |
| Equipment | 13,000.00 |
| Furniture | 15,000.00 |
| Management | 75,000.00 |
| Marketing | 30,666.67 |
| Navigation | 15,000.00 |
| Personnel | 35,000.00 |
| Purchasing | 49,500.00 |
| Recreation | 15,000.00 |

### 19. Find, for each department on the second floor, the average salary of the employees

A slightly more complex join and average

```
SELECT Department.DepartmentName,
            FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
    FROM Employee NATURAL JOIN Department
    WHERE DepartmentFloor = 2
    GROUP BY Department.DepartmentName;
```

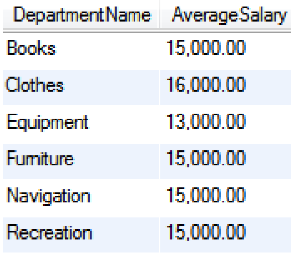


| DepartmentName | AverageSalary |
|---|---|
| Clothes | 16,000.00 |
| Recreation | 15,000.00 |

### 20. Find, for each department that sells items of type E, the average salary of the employees

Not a hard query, just lots of stuff in it – joins, conditions and grouping

```
SELECT Department.DepartmentName,
          FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
    FROM Employee INNER JOIN Department INNER JOIN Sale INNER JOIN Item
        ON Employee.DepartmentID = Department.DepartmentID
        AND Department.DepartmentID = Sale.DepartmentID
        AND Sale.ItemID = Item.ItemID
    WHERE ItemType = 'E'
    GROUP BY Department.DepartmentName;
```



| DepartmentName | AverageSalary |
|---|---|
| Books | 15,000.00 |
| Clothes | 16,000.00 |
| Equipment | 13,000.00 |
| Furniture | 15,000.00 |
| Navigation | 15,000.00 |
| Recreation | 15,000.00 |

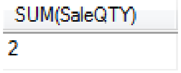
### 21.

(removed due to poorly structured question)

# Week 5

## 22. Find the total number of items of type E sold by the departments on the second floor

Similar to query 20 but a sum

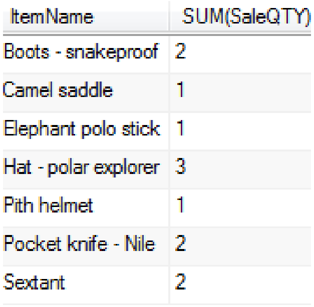


| SUM(SaleQTY) |
|---|
| 2 |

```
    SELECT SUM(SaleQTY) FROM Item INNER JOIN Sale INNER JOIN Department
       ON Item.ItemID = Sale.ItemID
       AND Department.DepartmentID = Sale.DepartmentID
       WHERE ItemType = 'E'
       AND DepartmentFloor = 2;
```

## 23. Find, for each item, the total quantity sold by the departments on the second floor

Similar to query 20 again…

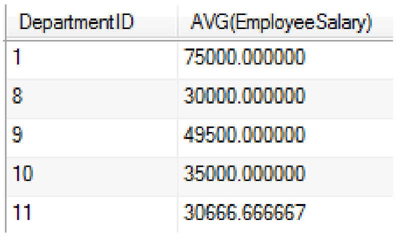


| ItemName | SUM(SaleQTY) |
|---|---|
| Boots - snakeproof | 2 |
| Camel saddle | 1 |
| Elephant polo stick | 1 |
| Hat - polar explorer | 3 |
| Pith helmet | 1 |
| Pocket knife - Nile | 2 |
| Sextant | 2 |

```
   SELECT Item.ItemName, SUM(SaleQTY) FROM Item
          INNER JOIN Sale INNER JOIN Department
      ON Item.ItemID = Sale.ItemID
      AND Department.DepartmentID = Sale.DepartmentID
      WHERE DepartmentFloor = 2
      GROUP BY Item.ItemName;
```

## 24. List the departments having an average salary over $25,000

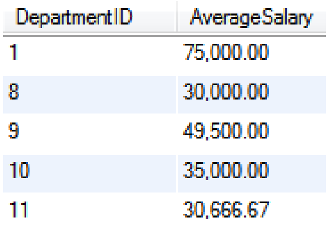
```
        SELECT DepartmentID, AVG(EmployeeSalary)
        FROM Employee GROUP BY DepartmentID
                HAVING AVG(EmployeeSalary) > 25000;
```



| DepartmentID | AVG(EmployeeSalary) |
|---|---|
| 1 | 75000.000000 |
| 8 | 30000.000000 |
| 9 | 49500.000000 |
| 10 | 35000.000000 |
| 11 | 30666.666667 |

This output looks poor… We can fix this using the Format function…

```
        SELECT DepartmentID,
                FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
             FROM Employee
             GROUP BY DepartmentID
                HAVING AVG(EmployeeSalary) > 25000;
```



| DepartmentID | AverageSalary |
|---|---|
| 1 | 75,000.00 |
| 8 | 30,000.00 |
| 9 | 49,500.00 |
| 10 | 35,000.00 |
| 11 | 30,666.67 |

Other useful formatting functions include UPPER, LOWER, LEFT, RIGHT

## 25. List the employees in the Accounting department and the difference between their salaries and the average salary of the department

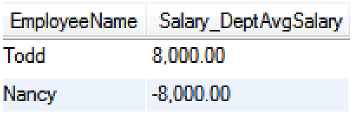
Need to get the average salary for Accounting employees in the same row as the department salary data.

Often due to the complexity of the queries it is often a good idea to create a View, to simplify the query. This has the advantage that parts of the query can be re-used.

First, determine the average salary of the employees in all departments and save this as a view.
Then, join this view to the Employee table matching the Accounting department's name.

```
CREATE VIEW vAvgSalaryDept (DepartmentID, dpavgsal) AS
    SELECT DepartmentID, AVG(EmployeeSalary) FROM Employee
        GROUP BY DepartmentID;

SELECT EmployeeName, FORMAT((EmployeeSalary - dpavgsal),2) AS Salary_DeptAvgSalary
    FROM vAvgSalaryDept NATURAL JOIN Employee NATURAL JOIN Department
        WHERE Department.DepartmentName = 'Accounting';
```

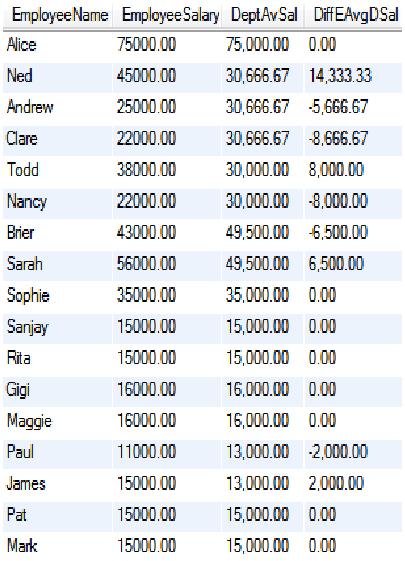


| EmployeeName | Salary_DeptAvgSalary |
|---|---|
| Todd | 8,000.00 |
| Nancy | -8,000.00 |

## 26. 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

We already have the average salaries of departments in a View (see q 25).
So we just need to use this and join it to department to get our answer.

```
SELECT EmployeeName, EmployeeSalary, FORMAT(dpavgsal,2) AS DeptAvSal,
            FORMAT(EmployeeSalary - dpavgsal,2) AS DiffEAvgDSal
        FROM vAvgSalaryDept NATURAL JOIN Employee
            WHERE vAvgSalaryDept.DepartmentID =
                        Employee.DepartmentID;
```



| EmployeeName | EmployeeSalary | DeptAvSal | DiffEAvgDSal |
|---|---|---|---|
| Alice | 75000.00 | 75,000.00 | 0.00 |
| Ned | 45000.00 | 30,666.67 | 14,333.33 |
| Andrew | 25000.00 | 30,666.67 | -5,666.67 |
| Clare | 22000.00 | 30,666.67 | -8,666.67 |
| Todd | 38000.00 | 30,000.00 | 8,000.00 |
| Nancy | 22000.00 | 30,000.00 | -8,000.00 |
| Brier | 43000.00 | 49,500.00 | -6,500.00 |
| Sarah | 56000.00 | 49,500.00 | 6,500.00 |
| Sophie | 35000.00 | 35,000.00 | 0.00 |
| Sanjay | 15000.00 | 15,000.00 | 0.00 |
| Rita | 15000.00 | 15,000.00 | 0.00 |
| Gigi | 16000.00 | 16,000.00 | 0.00 |
| Maggie | 16000.00 | 16,000.00 | 0.00 |
| Paul | 11000.00 | 13,000.00 | -2,000.00 |
| James | 15000.00 | 13,000.00 | 2,000.00 |
| Pat | 15000.00 | 15,000.00 | 0.00 |
| Mark | 15000.00 | 15,000.00 | 0.00 |

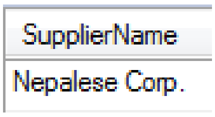
## 27. List the suppliers that deliver all the items of type B to departments on the second floor who sell all the items of type R

In this case we'll create a View showing departments on the 2nd floor who sell all items of type R.

Then we'll use the view to report on the suppliers that deliver all the items of type B to the departments identified in the view.

### 27.1. first, how would we do this *without* using a View…

```
SELECT SupplierName FROM Supplier WHERE NOT EXISTS
        (SELECT * FROM Item
            WHERE ItemType = 'B' AND NOT EXISTS
            (SELECT * FROM Delivery
                WHERE Delivery.ItemID = Item.ItemID
                AND Delivery.SupplierID = Supplier.SupplierID
                AND DepartmentID IN
                    (SELECT DepartmentID FROM Department
                        WHERE DepartmentFloor = 2 AND NOT EXISTS
                            (SELECT * FROM Item
                                WHERE ItemType = 'R' AND NOT EXISTS
                                    (SELECT * FROM Sale
                                    WHERE Sale.ItemID = Item.ItemID
                                    AND Sale.DepartmentID = Department.DepartmentID)))));
```

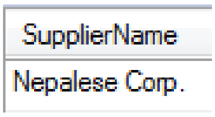


| SupplierName |
|---|
| Nepalese Corp. |

### 27.2. now, using A View…

```
CREATE VIEW vDeptFloor2ItemB AS
    (SELECT DepartmentID FROM Department
        WHERE DepartmentFloor = 2 AND NOT EXISTS
            (SELECT * FROM Item WHERE ItemType = 'R' AND NOT EXISTS
                    (SELECT * FROM Sale WHERE Sale.ItemID = Item.ItemID
                        AND Sale.DepartmentID = Department.DepartmentID)));
```

```
SELECT SupplierName FROM Supplier WHERE NOT EXISTS
        (SELECT * FROM Item WHERE ItemType = 'B' AND NOT EXISTS
                (SELECT * FROM Delivery WHERE Delivery.ItemID = Item.ItemID
                    AND Delivery.SupplierID = Supplier.SupplierID
                    AND DepartmentID IN
                        (SELECT DepartmentID FROM vDeptFloor2ItemB)));
```



| SupplierName |
|---|
| Nepalese Corp. |

## 28. List the suppliers that deliver all the items of type B to the departments that also sell all the items of type N

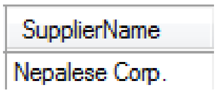
Split the problem into 3 parts:
Identify the departments that sell all items of type N – that's the 1st view.

Then select the departments to which all items of type B are delivered – that's the 2nd view.

Finally find the suppliers that supply all items of type B to the departments that appear in both views.

```
CREATE VIEW v33a AS
  (SELECT DepartmentID FROM Department WHERE NOT EXISTS
      (SELECT * FROM Item
        WHERE ItemType = 'N' AND NOT EXISTS
          (SELECT * FROM Sale
            WHERE Sale.DepartmentID = Department.DepartmentID
            AND Sale.ItemID = Item.ItemID)));

CREATE VIEW v33b AS
  (SELECT DepartmentID FROM Department WHERE NOT EXISTS
    (SELECT * FROM Item
      WHERE ItemType = 'B' AND NOT EXISTS
        (SELECT * FROM Delivery
          WHERE Delivery.DepartmentID = Department.DepartmentID
          AND Delivery.ItemID = Item.ItemID)));
```



| SupplierName |
|---|
| Nepalese Corp. |

```
SELECT SupplierName FROM Supplier WHERE NOT EXISTS
  (SELECT * FROM Item
    WHERE ItemType = 'B' AND NOT EXISTS
      (SELECT * FROM Delivery
        WHERE Delivery.SupplierID = Supplier.SupplierID
        AND Delivery.ItemID = Item.ItemID
        AND EXISTS
          (SELECT * FROM v33a
            WHERE Delivery.DepartmentID = v33a.DepartmentID)
        AND EXISTS
          (SELECT * FROM v33b
            WHERE Delivery.DepartmentID = v33b.DepartmentID)));
```

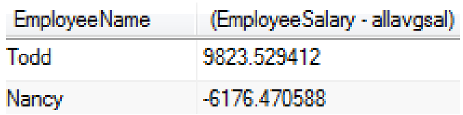
## 29. List the employees in the Accounting department and the difference between their salaries and the average salary of all of the departments

You can answer this with a View or a Subquery.
We'll demonstrate the View here and let you try the Subquery.

Using similar logic to query 25…

```
CREATE VIEW vAvgSalary(allavgsal) AS
    SELECT AVG(EmployeeSalary) FROM Employee;
```

```
SELECT EmployeeName, (EmployeeSalary - allavgsal)
    FROM vAvgSalary NATURAL JOIN Employee
    WHERE DepartmentID IN
        (SELECT DepartmentID FROM Department
            WHERE DepartmentName = 'Accounting');
```



| EmployeeName | (EmployeeSalary - allavgsal) |
|---|---|
| Todd | 9823.529412 |
| Nancy | -6176.470588 |

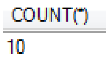
## 30. What is the number of supplier-department pairs in which the supplier delivers at least one item of type E to the department?

Again you can answer this with a View or a Subquery.

First, the view:
Set up a view that lists all the supplier-department pairs. Remove duplicates using Distinct.
Then use a count to get the answer.

```
CREATE VIEW v60 AS
    (SELECT DISTINCT SupplierID, DepartmentID FROM Delivery NATURAL JOIN Item
        WHERE ItemType = 'E' );

SELECT COUNT(*) FROM v60;
```



| COUNT(*) |
|---|
| 10 |

Now, using a subquery:

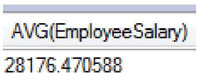
```
SELECT count(*) FROM
    (SELECT DISTINCT SupplierID, DepartmentID FROM Delivery NATURAL JOIN Item
        WHERE ItemType = 'E' ) AS tempname;
```

# Week 6

## 31. Give the overall average of the salaries in all departments

A really simple Average query

```
SELECT AVG(EmployeeSalary) FROM Employee;
```



| AVG(EmployeeSalary) |
|---|
| 28176.470588 |

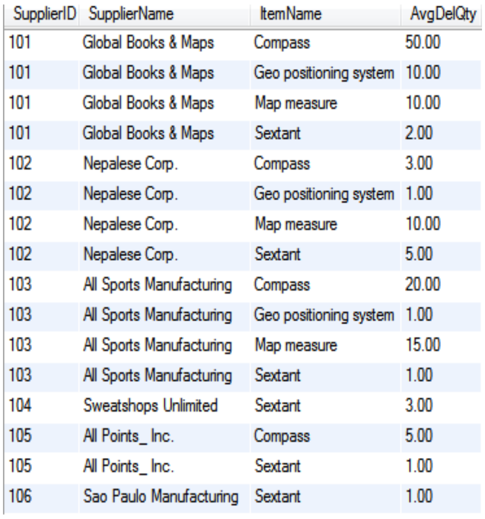
We could use the view we created of average salaries per department, but this gives a different answer due to rounding errors

```
SELECT AVG(dpavgsal) FROM vAvgSalaryDept;
```

## 32. What is the average delivery quantity of items of type N delivered by each company who delivers them

(similar to Query 45 )

```
SELECT Delivery.SupplierID, SupplierName, Item.ItemName,
         FORMAT(AVG(DeliveryQTY),2) AS AvgDelQty
   FROM Delivery INNER JOIN Supplier INNER JOIN Item
      ON Delivery.SupplierID = Supplier.SupplierID
      AND Delivery.ItemID = Item.ItemID
   WHERE ItemType = 'N'
   GROUP BY Delivery.SupplierID, SupplierName, Item.ItemName;
```

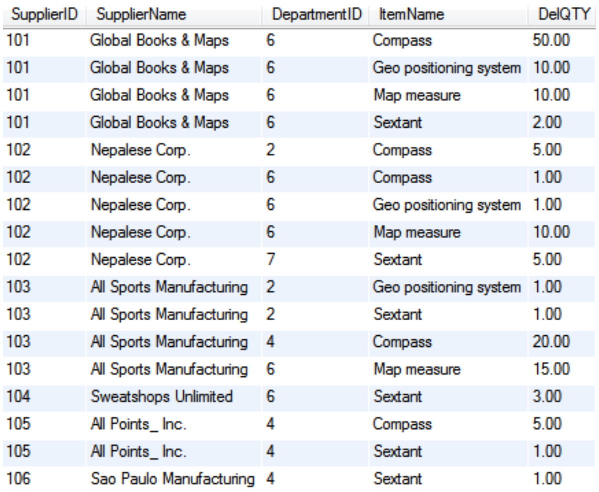


| SupplierID | SupplierName | ItemName | AvgDelQty |
|---|---|---|---|
| 101 | Global Books & Maps | Compass | 50.00 |
| 101 | Global Books & Maps | Geo positioning system | 10.00 |
| 101 | Global Books & Maps | Map measure | 10.00 |
| 101 | Global Books & Maps | Sextant | 2.00 |
| 102 | Nepalese Corp. | Compass | 3.00 |
| 102 | Nepalese Corp. | Geo positioning system | 1.00 |
| 102 | Nepalese Corp. | Map measure | 10.00 |
| 102 | Nepalese Corp. | Sextant | 5.00 |
| 103 | All Sports Manufacturing | Compass | 20.00 |
| 103 | All Sports Manufacturing | Geo positioning system | 1.00 |
| 103 | All Sports Manufacturing | Map measure | 15.00 |
| 103 | All Sports Manufacturing | Sextant | 1.00 |
| 104 | Sweatshops Unlimited | Sextant | 3.00 |
| 105 | All Points_ Inc. | Compass | 5.00 |
| 105 | All Points_ Inc. | Sextant | 1.00 |
| 106 | Sao Paulo Manufacturing | Sextant | 1.00 |

## 33. What is the average delivery quantity of items of type N delivered by each supplier to each department (for suppliers that deliver items of type N to the department)

Similar again to query 32, except we average on 3 levels (ie Group By)

```
SELECT Delivery.SupplierID, SupplierName, DepartmentID,
         Item.ItemName,
         FORMAT(AVG(DeliveryQTY),2) AS DelQTY
   FROM Delivery INNER JOIN Supplier INNER JOIN Item
      ON Delivery.SupplierID = Supplier.SupplierID
      AND Delivery.ItemID = Item.ItemID
   WHERE  ItemType = 'N'
   GROUP BY Delivery.SupplierID, SupplierName,
               DepartmentID,  Item.ItemName;
```



| SupplierID | SupplierName | DepartmentID | ItemName | DelQTY |
|---|---|---|---|---|
| 101 | Global Books & Maps | 6 | Compass | 50.00 |
| 101 | Global Books & Maps | 6 | Geo positioning system | 10.00 |
| 101 | Global Books & Maps | 6 | Map measure | 10.00 |
| 101 | Global Books & Maps | 6 | Sextant | 2.00 |
| 102 | Nepalese Corp. | 2 | Compass | 5.00 |
| 102 | Nepalese Corp. | 6 | Compass | 1.00 |
| 102 | Nepalese Corp. | 6 | Geo positioning system | 1.00 |
| 102 | Nepalese Corp. | 6 | Map measure | 10.00 |
| 102 | Nepalese Corp. | 7 | Sextant | 5.00 |
| 103 | All Sports Manufacturing | 2 | Geo positioning system | 1.00 |
| 103 | All Sports Manufacturing | 2 | Sextant | 1.00 |
| 103 | All Sports Manufacturing | 4 | Compass | 20.00 |
| 103 | All Sports Manufacturing | 6 | Map measure | 15.00 |
| 104 | Sweatshops Unlimited | 6 | Sextant | 3.00 |
| 105 | All Points_ Inc. | 4 | Compass | 5.00 |
| 105 | All Points_ Inc. | 4 | Sextant | 1.00 |
| 106 | Sao Paulo Manufacturing | 4 | Sextant | 1.00 |

## 34. Find the items sold by the departments on the second floor. (only show ItemID)

Note that in a real example we would want to also join to the item table and show the item name, as it is better understood by the users

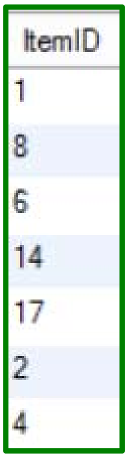
.

### 34.1. Attempt 1

```
SELECT DISTINCT ItemID FROM Sale INNER JOIN Department
        ON Department.DepartmentID = Sale.DepartmentID
        WHERE DepartmentFloor = 2;
```

### 34.2. Attempt 2

```
SELECT DISTINCT ItemID FROM Sale
        WHERE DepartmentID IN
                (SELECT DepartmentID FROM Department
                        WHERE DepartmentFloor = 2);
```



| ItemID |
|---|
| 1 |
| 8 |
| 6 |
| 14 |
| 17 |
| 2 |
| 4 |

### 34.3. Attempt 3

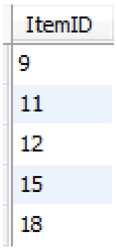
```
SELECT DISTINCT ItemID FROM Sale
        WHERE DepartmentID IN
                (SELECT DepartmentID FROM Department
                        WHERE Department.DepartmentID = Sale.DepartmentID
                                AND DepartmentFloor = 2);
```

### 34.4. Attempt 4

```
SELECT DISTINCT ItemID FROM Sale
        WHERE EXISTS
                (SELECT * FROM Department
                        WHERE Sale.DepartmentID = Department.DepartmentID
                                AND DepartmentFloor = 2);
```

## 35. Find the items sold by no department on the second floor

Compare this with query 2, which asked:
    “Which items are sold on the floors other than 2?”
(a slight but significant difference).



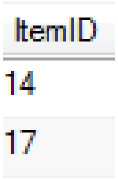
| ItemID |
|---|
| 9 |
| 11 |
| 12 |
| 15 |
| 18 |

```
SELECT DISTINCT ItemID FROM Sale
        WHERE ItemID NOT IN
        (SELECT DISTINCT ItemID FROM Sale NATURAL JOIN Department
                WHERE DepartmentFloor = 2);
```

## 36. Find the items sold by ALL departments on the second floor

The query is asking you to select the items from sales such that there does not exist a department on the $2^{nd}$ floor that does not sell the item.

```
SELECT Sale.ItemID FROM Sale NATURAL JOIN Department
        WHERE Department.DepartmentFloor = 2
        GROUP BY Sale.ItemID
                HAVING count(DISTINCT Department.DepartmentID) =
                        (SELECT count(DISTINCT DepartmentID) FROM Department
                                WHERE DepartmentFloor = 2);
```

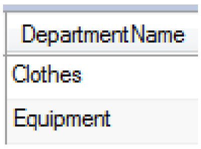


| ItemID |
|---|
| 14 |
| 17 |

## 37. Among all the departments with total salary greater than $25,000, find the departments that sell Item Number 17

Similar to query 9, we find the departments first, then do the selection from that list looking for Item 17

```
SELECT DISTINCT DepartmentName FROM Sale
    NATURAL JOIN Department D
    WHERE ItemID = 17
    AND D.DepartmentID IN
            (SELECT DepartmentID FROM Employee
               GROUP BY DepartmentID
                  HAVING SUM(EmployeeSalary) > 25000)
```

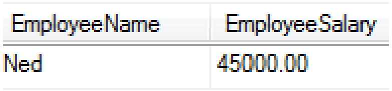


| DepartmentName |
|---|
| Clothes |
| Equipment |

## 38. Find the salary of Clare's manager

The inner (sub) query gets the manager, and the outer query gets the manager's details

```
SELECT EmployeeName, EmployeeSalary FROM Employee
        WHERE EmployeeID IN
                (SELECT BossID FROM Employee
                        WHERE EmployeeName = 'Clare');
```

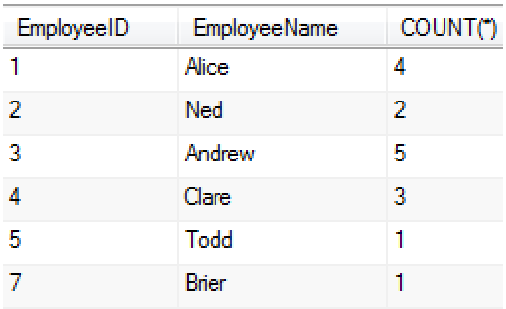


| EmployeeName | EmployeeSalary |
|---|---|
| Ned | 45000.00 |

## 39. Count the number of direct employees of each manager

Do a self join on the Employee table, group and count.

```
SELECT boss.EmployeeID, boss.EmployeeName, COUNT(*)
        FROM Employee wrk INNER JOIN Employee boss
                ON wrk.BossID = boss.EmployeeID
                GROUP BY boss.EmployeeID, boss.EmployeeName;
```

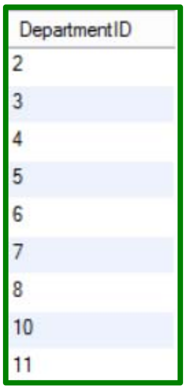


| EmployeeID | EmployeeName | COUNT(*) |
|---|---|---|
| 1 | Alice | 4 |
| 2 | Ned | 2 |
| 3 | Andrew | 5 |
| 4 | Clare | 3 |
| 5 | Todd | 1 |
| 7 | Brier | 1 |

## 40. Find the departments where all the employees earn less than their manager

Compare this to query 6, which asked:
"Which employees make more than their manager?"

```
        SELECT DISTINCT DepartmentID FROM Employee
                WHERE DepartmentID <> 1
                AND DepartmentID NOT IN
                (SELECT wrk.DepartmentID FROM Employee wrk INNER JOIN Employee boss
                                ON wrk.BossID = boss.EmployeeID
                                WHERE wrk.EmployeeSalary >= boss.EmployeeSalary);
```



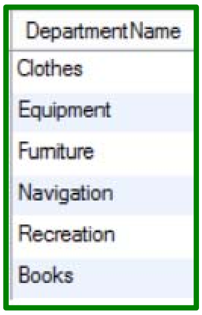
| DepartmentID |
|---|
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 10 |
| 11 |

# Week 7

## 41. List the departments selling items of type E that are delivered by Nepalese Corp. and/or which are sold by the Navigation department

This is similar to queries 15 and 16. We use one outer query to get the department name and two inner queries (separated by an OR) filter the departments.

```
SELECT DISTINCT DepartmentName FROM Sale NATURAL JOIN Department
    WHERE ItemID IN
        (SELECT Item.ItemID FROM Item INNER JOIN Delivery
                INNER JOIN Supplier
                ON Item.ItemID = Delivery.ItemID
                AND Delivery.SupplierID = Supplier.SupplierID
            WHERE SupplierName = 'Nepalese Corp.'
            AND ItemType = 'E')
    OR ItemID IN
        (SELECT ItemID FROM Sale NATURAL JOIN Department
            WHERE DepartmentName = 'Navigation');
```

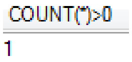


| DepartmentName |
|---|
| Clothes |
| Equipment |
| Furniture |
| Navigation |
| Recreation |
| Books |

## 42. Is it true that all the departments that sell items of type C are located on the third floor? (you should return either 1 for True or 0 for False)

There is no Boolean functionality in SQL.  Instead we can do a count, and test it with “> 0”, to give a 0 or 1 as an answer.

```
SELECT COUNT(*) > 0 FROM Department
    WHERE DepartmentFloor <> 3
    AND EXISTS
        (SELECT * FROM Sale NATURAL JOIN Item
            WHERE Sale.DepartmentID = Department.DepartmentID
            AND ItemType = 'C');
```

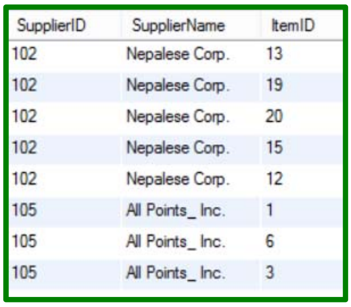


| COUNT(*)>0 |
|---|
| 1 |

## 43. List pairs of Supplier and Item where the supplier is the only deliverer of the item

For each of the items delivered, check to see that the supplier is the only deliverer of it.

```
SELECT DISTINCT Supplier.SupplierID, SupplierName, ItemID
    FROM Supplier NATURAL JOIN Delivery del1
    WHERE ItemID NOT IN
        (SELECT ItemID FROM Delivery
            WHERE Delivery.SupplierID <> del1.SupplierID);
```

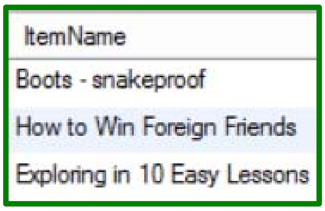


| SupplierID | SupplierName | ItemID |
|---|---|---|
| 102 | Nepalese Corp. | 13 |
| 102 | Nepalese Corp. | 19 |
| 102 | Nepalese Corp. | 20 |
| 102 | Nepalese Corp. | 15 |
| 102 | Nepalese Corp. | 12 |
| 105 | All Points_ Inc. | 1 |
| 105 | All Points_ Inc. | 6 |
| 105 | All Points_ Inc. | 3 |

## 44. List each item delivered to at least two departments by each supplier that delivers it

Inner query works out which item/supplier pairs only match1 department
Hint – first try: select ItemId, SupplierId, DepartmentId from delivery order by ItemId, SupplierId;.

```
SELECT DISTINCT ItemName
    FROM Delivery NATURAL JOIN Item
    WHERE ItemID NOT IN
        (SELECT ItemID FROM Delivery
            GROUP BY ItemID, SupplierID
                HAVING COUNT(DISTINCT DepartmentID) < 2);
```

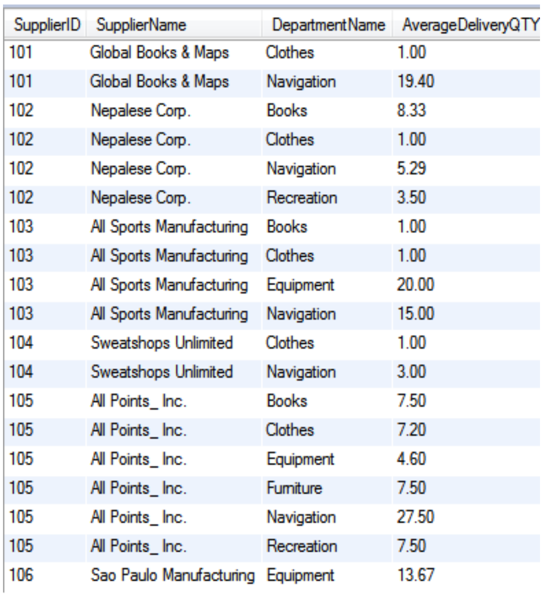


| ItemName |
|---|
| Boots - snakeproof |
| How to Win Foreign Friends |
| Exploring in 10 Easy Lessons |

## 45. What is, for each supplier, the average number of items that they deliver to each department?

Join tables using Group By to structure the result
(similar to query 32)

```
SELECT Delivery.SupplierID, SupplierName, DepartmentName,
        FORMAT(AVG(DeliveryQTY),2) AS AverageDeliveryQTY
    FROM Supplier NATURAL JOIN Delivery
        NATURAL JOIN Department
    GROUP BY Delivery.SupplierID, SupplierName,
            DepartmentName;
```

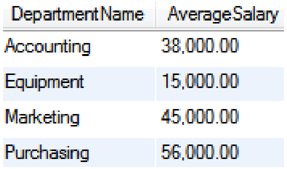


| SupplierID | SupplierName | DepartmentName | AverageDeliveryQTY |
|---|---|---|---|
| 101 | Global Books & Maps | Clothes | 1.00 |
| 101 | Global Books & Maps | Navigation | 19.40 |
| 102 | Nepalese Corp. | Books | 8.33 |
| 102 | Nepalese Corp. | Clothes | 1.00 |
| 102 | Nepalese Corp. | Navigation | 5.29 |
| 102 | Nepalese Corp. | Recreation | 3.50 |
| 103 | All Sports Manufacturing | Books | 1.00 |
| 103 | All Sports Manufacturing | Clothes | 1.00 |
| 103 | All Sports Manufacturing | Equipment | 20.00 |
| 103 | All Sports Manufacturing | Navigation | 15.00 |
| 104 | Sweatshops Unlimited | Clothes | 1.00 |
| 104 | Sweatshops Unlimited | Navigation | 3.00 |
| 105 | All Points_ Inc. | Books | 7.50 |
| 105 | All Points_ Inc. | Clothes | 7.20 |
| 105 | All Points_ Inc. | Equipment | 4.60 |
| 105 | All Points_ Inc. | Fumiture | 7.50 |
| 105 | All Points_ Inc. | Navigation | 27.50 |
| 105 | All Points_ Inc. | Recreation | 7.50 |
| 106 | Sao Paulo Manufacturing | Equipment | 13.67 |

## 46. For each department, find the average salary of the employees who earn more than the average salary of the department

The inner query averages the salaries of each department (see how it matches the departments).

```
SELECT DepartmentName, FORMAT(AVG(EmployeeSalary),2) AS AverageSalary
    FROM Employee Outt NATURAL JOIN Department
    WHERE EmployeeSalary >
        (SELECT AVG(EmployeeSalary) FROM Employee Inn
            WHERE Outt.DepartmentID = Inn.DepartmentID)
            GROUP BY DepartmentName;
```

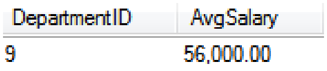


| DepartmentName | AverageSalary |
|---|---|
| Accounting | 38,000.00 |
| Equipment | 15,000.00 |
| Marketing | 45,000.00 |
| Purchasing | 56,000.00 |

## 47. List the departments where the average salary of the employees of each manager is more than $25,000

Getting harder now. We have to exclude manager's salaries from the calculation (this is accomplished by the inner query). We use Group By to get the data by department.

```
SELECT wrk.DepartmentID, FORMAT(AVG(wrk.EmployeeSalary),2) AS AvgSalary
  FROM Employee wrk
  WHERE wrk.EmployeeID NOT IN
     (SELECT Department.ManagerID FROM Department
        WHERE wrk.EmployeeID = Department.ManagerID
        AND wrk.DepartmentID = Department.DepartmentID)
           GROUP BY wrk.DepartmentID
           HAVING AVG(wrk.EmployeeSalary) > 25000;
```



| DepartmentID | AvgSalary |
|---|---|
| 9 | 56,000.00 |

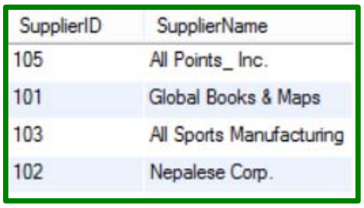
## 48. Find the suppliers that deliver both compasses and an item other than compasses

Attempt 1 uses the approach of finding those suppliers that supply things other than compasses, and also supply compasses (sub query)
Attempt 2 uses a more generalizable approach: Find suppliers that have delivered compasses and at least two items…(we use DISTINCT to handle multiple deliveries of compasses for the same supplier.

The generalizable approach is better as it allows queries such as "Find suppliers that deliver two items other than compasses" – change the >1 to >2 in the HAVING clause in Attempt 2 to do this.

### 48.1.Attempt 1

```
SELECT DISTINCT Delivery.SupplierID, SupplierName
  FROM Supplier NATURAL JOIN Delivery NATURAL JOIN Item
  WHERE ItemName <> 'Compass'
     AND Delivery.SupplierID IN
        (SELECT SupplierID FROM Delivery
            NATURAL JOIN Item
         WHERE ItemName = 'Compass');
```



| SupplierID | SupplierName |
|---|---|
| 105 | All Points_ Inc. |
| 101 | Global Books & Maps |
| 103 | All Sports Manufacturing |
| 102 | Nepalese Corp. |

### 48.2.Attempt 2

```
SELECT DISTINCT Delivery.SupplierID, SupplierName
  FROM Supplier NATURAL JOIN Delivery NATURAL JOIN Item
  WHERE Delivery.SupplierID IN
     (SELECT SupplierID FROM Delivery
            NATURAL JOIN Item
         WHERE ItemName = 'Compass')
  GROUP BY Delivery.SupplierID, SupplierName
     HAVING COUNT(DISTINCT ItemName) > 1;
```

Check your answer using:
select supplierid, itemid, itemname from Delivery natural join Item order by supplierid, itemid;

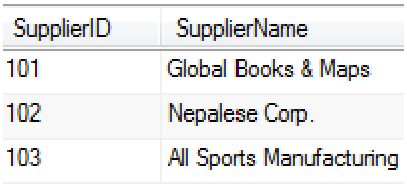
## 49. Who are the suppliers that deliver all the items of type N

This kind of query is called a Relational Divide. You use this when you need to find all of the rows of one table that are related to ALL of some other group of rows.

Here, we want to find any suppliers who deliver ALL of a particular subset of the Item table.

The usual way to do a relational divide in SQL is to use a subquery inside a subquery, each with a NOT EXISTS.

```
SELECT SupplierID, SupplierName FROM Supplier
    WHERE NOT EXISTS
        (SELECT * FROM Item WHERE ItemType = 'N'
            AND NOT EXISTS
                (SELECT * FROM Delivery
                    WHERE Delivery.SupplierID = Supplier.SupplierID
                    AND Delivery.ItemID = Item.ItemID));
```



| SupplierID | SupplierName |
|---|---|
| 101 | Global Books & Maps |
| 102 | Nepalese Corp. |
| 103 | All Sports Manufacturing |

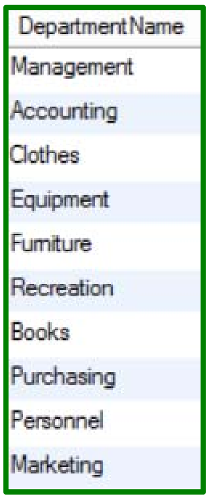
This is like saying: “Show each supplier for whom there is no item of type N for which there are no deliveries by that supplier of that item”.

Check your answer using: **SELECT * FROM Item WHERE ItemType = 'N';**
and **SELECT SupplierId, ItemId FROM Delivery order by SupplierId, ItemId;**

## 50. List the departments that have not recorded a sale for all the items of type N

First, use a Relational Divide to find the departments that have sold all items of type N.
Then use NOT IN to get those that *haven’t* sold all items of type N.

```
SELECT DepartmentName FROM Department
    WHERE DepartmentID NOT IN
        (SELECT DepartmentID FROM Department
            WHERE NOT EXISTS
                (SELECT * FROM Item
                    WHERE ItemType = 'N'
                    AND NOT EXISTS
                        (SELECT * FROM Sale
                            WHERE Sale.DepartmentID = Department.DepartmentID
                            AND Sale.ItemID = Item.ItemID)));
```



| DepartmentName |
|---|
| Management |
| Accounting |
| Clothes |
| Equipment |
| Furniture |
| Recreation |
| Books |
| Purchasing |
| Personnel |
| Marketing |

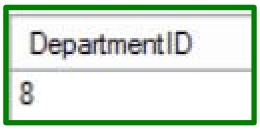
# Week 8 - CHALLENGE WEEK

## 51. List the departments that have at least one sale of all the items delivered to them

Another relational divide, or a variation on one, using 3 tables (in this case Sale and 2 copies of Delivery). The second version uses NOT IN as a replacement for NOT EXISTS.

### 51.1.Attempt 1

```
SELECT DISTINCT DepartmentID FROM Delivery del1
    WHERE NOT EXISTS
        (SELECT * FROM Delivery del2
            WHERE del2.DepartmentID = del1.DepartmentID
            AND NOT EXISTS
                (SELECT * FROM Sale
                    WHERE del2.ItemID = Sale.ItemID
                    AND del1.DepartmentID = Sale.DepartmentID));
```



| DepartmentID |
|---|
| 8 |

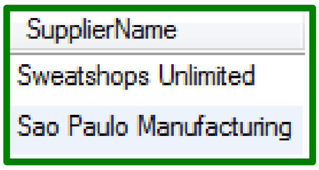
### 51.2.Attempt 2

```
SELECT DISTINCT DepartmentID FROM Delivery del1
    WHERE NOT EXISTS
        (SELECT * FROM Delivery del2
            WHERE del2.DepartmentID = del1.DepartmentID
            AND ItemID NOT IN
                (SELECT ItemID FROM Sale
                    WHERE DepartmentID = del1.DepartmentID));
```

## 52. List the suppliers that deliver only the items sold by the Books department

Select suppliers for which there does not exist a delivery that does not include the items sold by the Books department.

```
SELECT SupplierName FROM Supplier
    WHERE SupplierID IN
        (SELECT SupplierID FROM Delivery)
            AND NOT EXISTS
                (SELECT * FROM Delivery
                WHERE Delivery.SupplierID = Supplier.SupplierID
                    AND ItemID NOT IN
                        (SELECT ItemID FROM Sale NATURAL JOIN Department
                    WHERE DepartmentName = 'Books'));
```



| SupplierName |
|---|
| Sweatshops Unlimited |
| Sao Paulo Manufacturing |

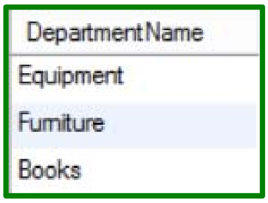
## 53.

Removed due to poorly structured question

## 54. List the departments for which each item delivered to the department is delivered to some other department as well

Find departments where there is not a delivery where a supplier does not deliver the item to some other department.

```
SELECT DISTINCT DepartmentName
    FROM Delivery del1 NATURAL JOIN Department
    WHERE NOT EXISTS
        (SELECT * FROM Delivery del2
            WHERE del2.DepartmentID = del1.DepartmentID
            AND ItemID NOT IN
                (SELECT ItemID FROM Delivery del3
                    WHERE del3.DepartmentID <> del1.DepartmentID));
```

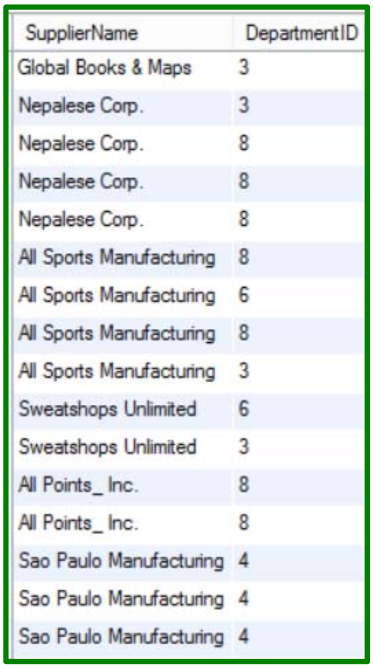


| DepartmentName |
|---|
| Equipment |
| Furniture |
| Books |

## 55. List the supplier-department pairs where the department sells all items delivered to it by the supplier

This is complex because you have to match the department name and the name of sales and deliveries.

```
SELECT SupplierName, DepartmentID
    FROM Delivery del1 NATURAL JOIN Supplier
    WHERE NOT EXISTS
        (SELECT * FROM Delivery
            WHERE Delivery.DepartmentID = del1.DepartmentID
            AND Delivery.SupplierID = del1.SupplierID
            AND ItemID NOT IN
                (SELECT ItemID FROM Sale
                    WHERE Sale.DepartmentID = del1.DepartmentID));
```

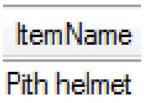


| SupplierName | DepartmentID |
|---|---|
| Global Books & Maps | 3 |
| Nepalese Corp. | 3 |
| Nepalese Corp. | 8 |
| Nepalese Corp. | 8 |
| Nepalese Corp. | 8 |
| All Sports Manufacturing | 8 |
| All Sports Manufacturing | 6 |
| All Sports Manufacturing | 8 |
| All Sports Manufacturing | 3 |
| Sweatshops Unlimited | 6 |
| Sweatshops Unlimited | 3 |
| All Points_ Inc. | 8 |
| All Points_ Inc. | 8 |
| Sao Paulo Manufacturing | 4 |
| Sao Paulo Manufacturing | 4 |
| Sao Paulo Manufacturing | 4 |

## 56. List the items delivered to all departments by all suppliers

This is a three part query: find the items delivered by all suppliers, find the items delivered to all departments and which items satisfy both conditions – joining the 1st two divides together. We exclude the non-sales departments in the 2nd divide (NOT IN (‘Management….’))

```
SELECT ItemName FROM Item
    WHERE NOT EXISTS
        (SELECT * FROM Supplier
            WHERE NOT EXISTS
                (SELECT * FROM Delivery
                    WHERE Delivery.ItemID = Item.ItemID
                    AND Delivery.SupplierID = Supplier.SupplierID))
    AND NOT EXISTS
        (SELECT * FROM Department
            WHERE DepartmentName
            NOT IN ('Management', 'Marketing', 'Personnel', 'Accounting', 'Purchasing')
            AND NOT EXISTS
                (SELECT * FROM Delivery
                    WHERE Delivery.ItemID = Item.ItemID
                    AND Delivery.DepartmentID = Department.DepartmentID));
```

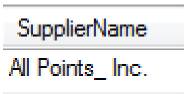


| ItemName |
|---|
| Pith helmet |

## 57. List the suppliers that deliver every item of type C to the same department on the second floor

Another divide… the inner query has three Where conditions to handle the “same department” requirement

```
SELECT SupplierName FROM Supplier
    WHERE EXISTS
        (SELECT * FROM Department WHERE DepartmentFloor = 2
            AND NOT EXISTS
                (SELECT * FROM Item WHERE ItemType = 'C'
                    AND NOT EXISTS
                        (SELECT * FROM Delivery
                            WHERE Delivery.SupplierID = Supplier.SupplierID
                            AND Delivery.ItemID = Item.ItemID
                            AND Delivery.DepartmentID = Department.DepartmentID)));
```

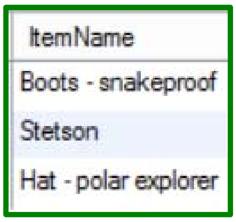


| SupplierName |
|---|
| All Points_ Inc. |

## 58. List the items that are delivered only by the suppliers that deliver something to all the departments

This one’s a bit tricky as only some department have deliveries so we have to get rid of the management departments in the query.

```
SELECT DISTINCT ItemName
    FROM Delivery del1 NATURAL JOIN Item
    WHERE NOT EXISTS
        (SELECT * FROM Delivery del2
            WHERE del2.ItemID = del1.ItemID
            AND SupplierID NOT IN
                (SELECT SupplierID FROM Delivery
                    GROUP BY SupplierID
                        HAVING COUNT(DISTINCT DepartmentID) =
                            (SELECT COUNT(*) FROM Department
                                WHERE DepartmentName NOT IN ('Management',
                                    'Marketing', 'Personnel','Accounting', 'Purchasing'))));
```

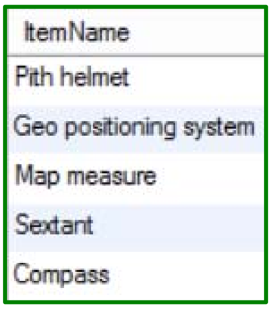


| ItemName |
|---|
| Boots - snakeproof |
| Stetson |
| Hat - polar explorer |

## 59. List the items delivered by every supplier that delivers all items of type N

This is a *double* divide…The inner query (shown inside the purple box) determines the suppliers that deliver all items of type N. The outer query determines which of the suppliers (returned by the inner divide) provide all these items.

```
SELECT DISTINCT ItemName FROM Delivery Del
    NATURAL JOIN Item
    WHERE NOT EXISTS
        (SELECT * FROM Supplier
            WHERE NOT EXISTS
                (SELECT * FROM Item
                    WHERE ItemType = 'N'
                    AND NOT EXISTS
                        (SELECT * FROM Delivery
                            WHERE Delivery.ItemID = Item.ItemID
                            AND Delivery.SupplierID = Supplier.SupplierID))
            AND NOT EXISTS
                (SELECT * FROM Delivery
                    WHERE Delivery.ItemID = Del.ItemID
                    AND Delivery.SupplierID = Supplier.SupplierID));
```

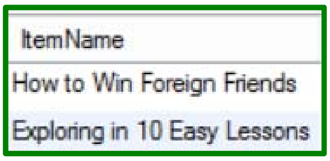


| ItemName |
|---|
| Pith helmet |
| Geo positioning system |
| Map measure |
| Sextant |
| Compass |

## 60. List the items sold only by departments that sell all the items delivered to them

Identify which items sold by departments that sell all items delivered to them
Then make sure no other department sells that item

```
SELECT DISTINCT ItemName FROM Sale sal NATURAL JOIN Item
    WHERE DepartmentID IN
        (SELECT DepartmentID FROM Department dept1
            WHERE NOT EXISTS
                (SELECT * FROM Delivery
                    WHERE Delivery.DepartmentID = dept1.DepartmentID
                    AND ItemID NOT IN
                        (SELECT ItemID FROM Sale
                            WHERE Sale.DepartmentID = dept1.DepartmentID)))
    AND NOT EXISTS
        (SELECT * FROM Sale
            WHERE ItemID = sal.ItemID
            AND DepartmentID NOT IN
                (SELECT DepartmentID FROM Department dept2
                    WHERE NOT EXISTS
                        (SELECT * FROM Delivery
                            WHERE Delivery.DepartmentID = dept2.DepartmentID
                            AND ItemID NOT IN
                                (SELECT ItemID FROM Sale
                                    WHERE Sale.DepartmentID = dept2.DepartmentID))));
```



| ItemName |
|---|
| How to Win Foreign Friends |
| Exploring in 10 Easy Lessons |

This is probably the longest query you will write.

# SQL Divide - How it works

“Find the items that have appeared in all sales”

pThis query could be phrased as: "Select item names where *forall* sales, there *exists* a lineitem row recording that this item was sold." A quick inspection of Table 1 shows that one item satisfies this condition (itemno = 2). While SQL does not directly support *forall*, it can be expressed using *exists*. The query becomes "Find items such that there does not exist a sale in which this item does not appear." The equivalent SQL expression is (this is called an SQL Divide):

```
SELECT itemno, itemname FROM item
   WHERE NOT EXISTS
      (SELECT * FROM sale
         WHERE NOT EXISTS
            (SELECT * FROM lineitem
               WHERE lineitem.itemno = item.itemno
               AND lineitem.saleno = sale.saleno))
```

| itemno | itemname |
|---|---|
| 2 | Pocket knife--Avon |

You may be convinced that this query is correct by walking through it step by step using the data in Table 1 and examining the outcome of each step of the query (see Table 2).

**Step 1.** The first row of item has itemno=1, and the first row of sale has saleno = 1. The innermost query becomes

```
SELECT * FROM lineitem
   WHERE lineitem.itemno = 1
   AND lineitem.saleno = 1
```

Examination of the lineitem table shows this query returns *false*, which is negated by the NOT before the innermost EXISTS to give *true*.

**Step 2.** We now advance to the second row of sale with saleno = 2, and execute the innermost query again. This query will also return false which will be negated to true.

**Steps 3-5.** The results of these and other steps are summarized in Table 2.

**Step 6.** The subquery

```
SELECT * FROM sale
    WHERE NOT EXISTS (...
```

has now been executed for each value of saleno for the first value of itemno and returned *true* for each row. Thus this query returns *true*, and the NOT before the EXISTS of this query changes the *true* to *false*. Since the value returned is *false*, the current value of itemno is not reported.

| sale |
|---|
| saleno |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

| lineitem | | |
|---|---|---|
| lineno | saleno | itemno |
| 1 | 1 | 2 |
| 1 | 2 | 6 |
| 2 | 2 | 16 |
| 3 | 2 | 19 |
| 4 | 2 | 2 |
| 1 | 3 | 4 |
| 2 | 3 | 2 |
| 1 | 4 | 4 |
| 2 | 4 | 9 |
| 3 | 4 | 13 |
| 4 | 4 | 14 |
| 5 | 4 | 3 |
| 6 | 4 | 2 |
| 1 | 5 | 10 |
| 2 | 5 | 11 |
| 3 | 5 | 12 |
| 4 | 5 | 13 |
| 5 | 5 | 2 |

| item |
|---|
| itemno |
| 1 |
| 2 |
| 3 |
| ... |
| |

Table 1: Condensed versions of sale, lineitem, and item

**Steps 7-11.** The results of these steps are shown in Table 2.

**Step 12.** The subquery

```
SELECT * FROM sale
   WHERE NOT EXISTS (...
```

has now been executed for each value of saleno for the second value of itemno and returned *false* for each row. Although this query returns *false*, the NOT before the EXISTS changes the *false* to *true*. Since the value returned is *true*, the current value of itemno is reported. Item 2 appears in all sales.

It will take you some time to step through this example to understand how the double **NOT EXISTS** works. Once you do, you will be convinced that the query does work.

| **Step** | **itemno** | **saleno** | **SELECT * FROM** lineitem **WHERE** lineitem.itemno = item.itemno **AND** lineitem.saleno = sale.saleno | **NOT EXISTS** | **SELECT** * **FROM** sale **WHERE NOT EXISTS (SELECT * FROM** lineitem **WHERE** lineitem.itemno = item.itemno **AND** lineitem.saleno =sale.saleno) | **NOT EXISTS** |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | false | true | | |
| 2 | 1 | 2 | false | true | | |
| 3 | 1 | 3 | false | true | | |
| 4 | 1 | 4 | false | true | | |
| 5 | 1 | 5 | false | true | | |
| 6 | | | | | true | false |
| 7 | 2 | 1 | true | false | | |
| 8 | 2 | 2 | true | false | | |
| 9 | 2 | 3 | true | false | | |
| 10 | 2 | 4 | true | false | | |
| 11 | 2 | 5 | true | false | | |
| 12 | | | | | false | true |
| 13 | 3 | 1 | false | true | | |
| 14 | 3 | 2 | false | true | | |
| 15 | 3 | 3 | false | true | | |
| 16 | 3 | 4 | true | false | | |
| 17 | 3 | 5 | false | true | | |
| 18 | | | | | true | false |

Table 2: Results of the first 18 steps of the divide statement.

# Section 4: Write your own SQL queries

Write your own queries to answer these questions. Answers will be made available to you around week 8. These questions use the same colour code:

- green –easy
- blue - moderate
- red - hard

## Week 3

1. List the green items of type C.
2. Find the departments that sell at least 4 items
3. Find the departments that sell at least 4 items and list how many items each department sells
4. Find the employees who are in the same department as their manager.
5. Find the names of brown items sold by the Recreation department.
6. Find the employees whose salary is less than half that of their managers.

## Week 4

7. Find the departments that have never sold a geo positioning system
8. Find the items sold by at least two departments.
9. Find the name of the highest-paid employee in the Marketing department.
10. Find the names of employees who make 40 per cent less than the average salary.
11. Find the number of employees with a salary under $15,000.
12. Find the number of units sold of each item.
13. Find the supplier that delivers no more than two items

## Week 5

14. Find the suppliers that have never delivered a compass.
15. Find, for each department, its floor and the average salary in the department.
16. List the departments on the second floor.

17. **List the names of items delivered by each supplier. Arrange the report by supplier name, and within supplier name, list the items in alphabetical order.**

18. **List the number of employees in each department.**

19. **Whom does Todd manage?**

20. **Find the first name of Sophie's boss.**

## Week 6

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

22. **Who earns the lowest salary?**

23. **Of those items delivered, find the items not delivered to the Books department.**

24. **Find the departments that have sold compasses and at least six other items.**

25. **Find the brown items sold by no department on the second floor.**

26. **Find the items delivered by all suppliers**

27. **Find the items not delivered by Nepalese Corp**

## Week 7

28. **Find the items delivered for which there have been no sales.**

29. **Find the names of employees with a salary greater than the minimum salary paid to a manager.**

30. **Find the names of suppliers that do not supply compasses or geo positioning systems.**

31. **Find the number of items of type C sold by the departments on the third floor.**

32. **If Nancy's boss has a boss, who is it?**

33. **List each employee and the difference between his or her salary and the average salary of his or her department.**

34. **List the departments on the second floor that contain more than one employee.**

## Week 8

35. **List the names of employees who earn more than the average salary of employees in the Accounting department.**

36. **List the names of managers who supervise only one person.**

37. **List the names of employees who earn less than the minimum salary of the Marketing department.**

38. **List the department and the item where the department is the only seller of that item.**

39. **Which department has the highest average salary?**

40. **Find the khaki items delivered by all suppliers.**

41. **Find the suppliers that deliver to all departments. Don't forget to exclude the administrative departments, which don't sell items.**

42. **Find the items delivered to all departments except Administration departments (Management, Marketing, Personnel, Accounting, Purchasing).**