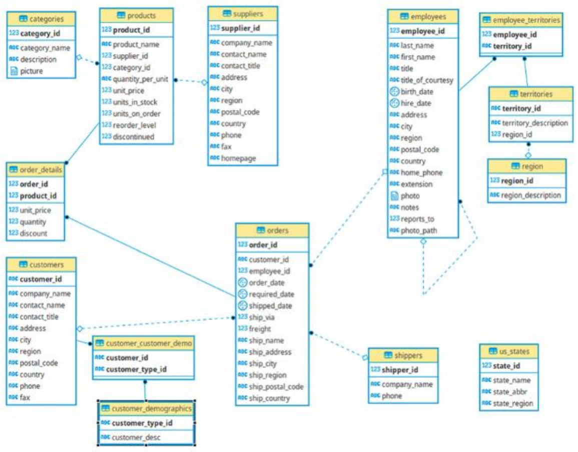
1) Northwind Queries

1.1

```sql
-- Returns the Required Column Fields for the Table:
SELECT  c.CustomerID AS "Customer ID",
        c.CompanyName AS "Company Name",
        c.Address,
        c.City,
        c.Country
FROM    Customers c
-- Conditions to return:
WHERE   c.City = 'Paris'
OR      c.City = 'London'
```



1.2

```sql
-- Returns the Required Column Field for the Table:
SELECT  p.ProductName AS "Products stored in Bottles"
FROM    Products p
-- Conditions to return:
-- Inspecting the field text to contain the phrase "bottles"
WHERE   p.QuantityPerUnit LIKE '%bottles%';
```

1.3

```sql
-- Returns the Required Column Fields for the Table:
SELECT      p.ProductName AS "Products stored in Bottles",
            s.CompanyName,
            s.Country
FROM        Products p
-- Link the Suppliers table to the Products Table for cross-referencing:
INNER JOIN  Suppliers s ON s.SupplierID = p.SupplierID
-- Conditions to return:
-- Inspecting the field text to contain the phrase "bottles"
WHERE       p.QuantityPerUnit LIKE '%bottles%';
```

1.4

```sql
-- Returns the Required Column Fields for the Table:
SELECT      c.CategoryName AS "Category",
            COUNT(*) AS "Number of Products"
FROM        Products p
-- Link the Categories table to the Products Table for cross-referencing:
INNER JOIN  Categories c ON c.CategoryID = p.CategoryID
GROUP BY    c.CategoryName
ORDER BY    "Number of Products" DESC
```

1.5

```
-- Returns the Required Column Fields for the Table:
-- Concatenating the Title, FirstName and LastName into 1 column
SELECT  CONCAT(e.TitleOfCourtesy, ' ', e.FirstName, ' ', e.LastName) AS "Employee Name",
        e.City
FROM    Employees e
```

1.6

```
-- Returns the Required Column Fields for the Table:
SELECT      t.RegionID,
            -
- Rounds the value returned to 2 decimal places and prevents the return of successive 0 val
ues
            FORMAT(ROUND(SUM(od.UnitPrice*od.Quantity*(1-
od.Discount)), 2), '##.##') AS "Total Sales"
FROM        [Order Details] od
-- Link the orders table to the [Order Details] Table for cross-referencing:
INNER JOIN  orders o ON o.OrderID = od.OrderID
-- Link the Employees table to the orders Table for cross-referencing:
INNER JOIN  Employees e ON e.EmployeeID = o.EmployeeID
-- Link the EmployeeTerritories table to the Employees Table for cross-referencing:
INNER JOIN  EmployeeTerritories et ON et.EmployeeID = e.EmployeeID
-- Link the Territories table to the EmployeeTerritories Table for cross-referencing:
INNER JOIN  Territories t ON t.TerritoryID = et.TerritoryID
GROUP BY    t.RegionID
-- Conditions in order to return:
HAVING      SUM(od.UnitPrice*od.Quantity*(1-od.Discount)) >= 1000000
ORDER BY    t.RegionID
```

1.7

```
-- Returns the Required Column Field for the Table:
SELECT  Count(*) AS "Orders which have Freight > 100 from UK or USA"
FROM    Orders o
-- Conditions in order to return:
WHERE   o.Freight > 100
AND     o.ShipCountry IN('UK', 'USA')
```

1.8

```
-- Returns the Required Column Field for the Table:
SELECT TOP 1    od.OrderID AS "Order Number with Highest (Value) of Discount"
FROM            [Order Details] od
GROUP BY        od.OrderID
ORDER BY        SUM(od.Discount*od.UnitPrice*od.Quantity) DESC
```

2) Spartans Table

2.1

```
-- Creates a table for the spartan data:
CREATE TABLE spartan_table
(
    spartan_ID INT IDENTITY(1,1) PRIMARY KEY,
    spartan_title VARCHAR(8),
    spartan_firstName VARCHAR(20),
    spartan_lastName VARCHAR(20),
    spartan_university VARCHAR(40),
    spartan_university_course VARCHAR(50),
    spartan_university_mark CHAR(3),
);
-- Outputs the table for inspection
select * FROM spartan_table
```

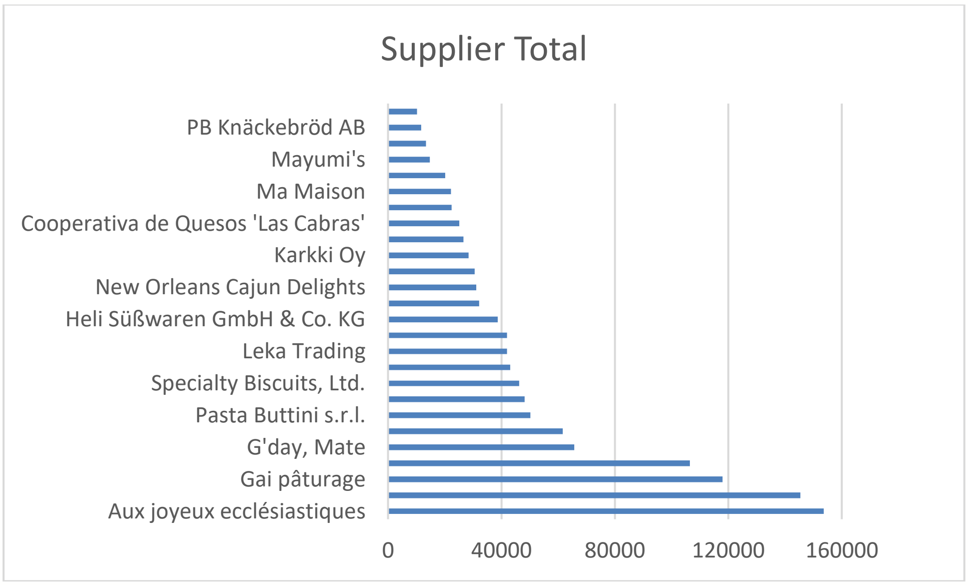
2.2

```
INSERT INTO spartan_table
(
    spartan_title,
    spartan_firstName,
    spartan_lastName,
    spartan_university,
    spartan_university_course,
    spartan_university_mark
)
VALUES
(
    'Mr', 'Alasdair', 'Malcolm', 'Exeter', 'Electronic Engineering', '2:2'
),
(
    'Mr', 'Jakub', 'Matyjewicz', 'Poznan University of Technology', 'Technical Physics', 'N
/A'
),
(
    'Mr', 'Alex', 'Sikorski', 'University of Coding', 'Coding', '1st'
),
(
    'Mr', 'Golam', 'Choudhury', 'City University of London', 'Aeronautical Engineering', '2
:1'
),
(
    'Mr', 'Matthew', 'Holmes', 'University of Bath', 'Computer Science and Mathematics', '2
:2'
);

-- Outputs the table for inspection
select * FROM spartan_table
```

3) Northwind Data Analysis Linked to Excel

3.1
```
-- Returns the Required Column Fields for the Table:
SELECT      CONCAT(e1.TitleOfCourtesy, ' ', e1.FirstName, ' ', e1.LastName) AS "Employee Na
me",
            CONCAT(e2.TitleOfCourtesy, ' ', e2.FirstName, ' ', e2.LastName) AS "Reports To:
"
FROM        Employees e1
-- Link the Employees table to a copy of itself for cross-referencing:
LEFT JOIN   Employees e2 ON e1.ReportsTo = e2.EmployeeID
WHERE       e2.FirstName IS NOT NULL
AND         e2.LastName IS NOT NULL;
/*  I'm not sure how it should be handled here (Dr Fuller being 'the boss' and so reports t
o no one / NULL)
    For now I have just used the LEFT JOIn to remove the majority of NULL values and the WH
ERE/AND clauses to prevent the following issue:

    When a NULL Title/FirstName/LastName is present, the CONCAT function effectively create
s "  ", rather than NULL*/
```

3.2
```
-- Returns the Required Column Fields for the Table:
SELECT DISTINCT     supp.CompanyName AS "Company Name",
                    ROUND(SUM(ordet.UnitPrice*ordet.Quantity*(1-
Discount)), 2) AS "Total Sales"
FROM                [Order Details] ordet
-- Link the Products table to the [Order Details] Table for cross-referencing:
INNER JOIN          Products prod ON prod.ProductID = ordet.ProductID
-- Link the Suppliers table to the Products Table for cross-referencing:
INNER JOIN          Suppliers supp ON supp.SupplierID = prod.SupplierID
GROUP BY            supp.CompanyName
-- Conditions to return:
HAVING              SUM(ordet.UnitPrice*ordet.Quantity*(1-Discount)) >= 10000
ORDER BY            "Total Sales" DESC
```

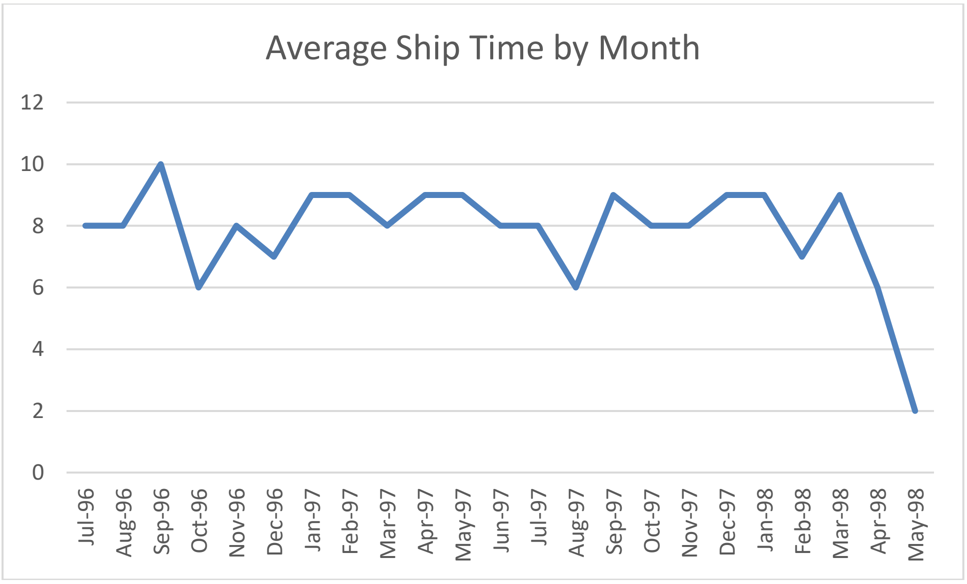


3.3

```
-- Returns the Required Column Fields for the Table:
SELECT TOP 10   cust.CompanyName AS "Company Name",
                -- Converts the 'total value' into the US-
dollar currancy format for display
                FORMAT(SUM(ordet.UnitPrice*ordet.Quantity), 'c2') AS "Total Value of Orders
 Shipped"
FROM            Orders ord
-- Link the Customers table to the Orders table for cross-referencing:
INNER JOIN      Customers cust ON cust.CustomerID = ord.CustomerID
-- Link the [Order Details] table to the Orders table for cross-referencing:
INNER JOIN      [Order Details] ordet ON ordet.OrderID = ord.OrderID
-- Selects the YTD records (1st Jan -> Present):
WHERE           DATEDIFF(year, ord.OrderDate, (SELECT MAX(FORMAT(ord.OrderDate, 'yyyy'))
                                               FROM Orders ord)) = 0

/*  WHERE DATEDIFF(day, ord.ShippedDate,   (SELECT TOP 1 FORMAT(ord.ShippedDate, 'yyyy-MM-
dd')
                                          FROM Orders ord
                                          WHERE ord.ShippedDate IS NOT NULL
                                          ORDER BY ord.ShippedDate DESC )) <= 365.25
     */
/*  This snippet was my original understanding of 'YTD', however I now believe that it is
    meaning "during the current calendar year" (Jan->Now), rather than (365 days ago -
> Now)    */

AND             ord.ShippedDate IS NOT NULL
GROUP BY        cust.CompanyName
ORDER BY        SUM(ordet.UnitPrice*ordet.Quantity) DESC
```

3.4

```
-- Returns the Required Column Fields for the Table:
            -- Displays the date format e.g "Jan-98"
SELECT      MAX(FORMAT(ord.OrderDate, 'MMM-yy')) AS "Month",
            AVG(DATEDIFF(day, ord.OrderDate, ord.ShippedDate)) AS "Average Ship Time"
FROM        orders ord
GROUP BY    FORMAT(ord.OrderDate, 'yyyy-MM')
ORDER BY    MAX(FORMAT(ord.OrderDate, 'yyyy-MM'))
```


Average Ship Time by Month
Jul-96
Aug-96
Sep-96
Oct-96
Nov-96
Dec-96
Jan-97
Feb-97
Mar-97
Apr-97
May-97
Jun-97
Jul-97
Aug-97
Sep-97
Oct-97
Nov-97
Dec-97
Jan-98
Feb-98
Mar-98
Apr-98
May-98