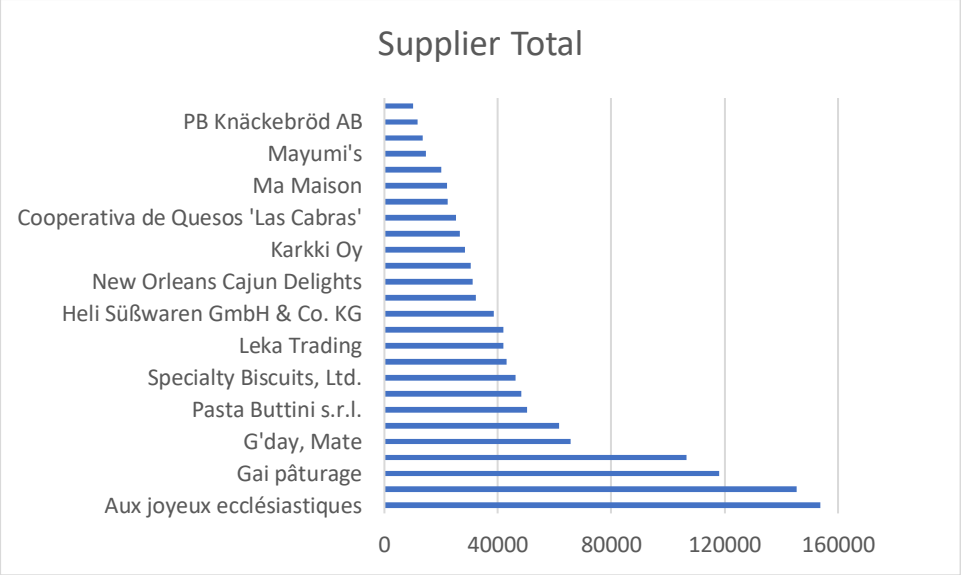
```sql
--QUESTION 1

USE Northwind;

--1.1
SELECT c.CustomerID,c.CompanyName,c.Address,c.City,c.Country--Column picked
FROM Customers c --From Customers table
WHERE c.City='Paris' OR c.City='London'--Limitting data to paris and london

--1.2
SELECT p.ProductName
FROM Products p
WHERE p.QuantityPerUnit LIKE '%bottle%'--
limiting it so only the units with blottle inside it is shown

--1.3
SELECT p.ProductName,s.CompanyName AS "Supplier",s.Country
FROM Products p LEFT JOIN Suppliers s--join is used to allow company name
ON p.SupplierID=s.SupplierID
WHERE p.QuantityPerUnit LIKE '%bottle%'--
limiting it so only the units with blottle inside it is shown

--1.4
SELECT c.CategoryName,COUNT(c.CategoryID) AS "Total product"
FROM Categories c INNER JOIN Products p ON c.CategoryID=p.CategoryID
GROUP BY c.CategoryName,c.CategoryID
ORDER BY "Total product" DESC

--1.5
SELECT e.EmployeeID,e.TitleOfCourtesy+e.FirstName+' '+e.LastName AS "UK Employees"
FROM Employees e
WHERE e.Country='UK'

--1.6
SELECT t.RegionID,ROUND(SUM(od.Quantity*od.UnitPrice*(1-od.Discount)),2) AS "Total Sales"
FROM Territories t INNER JOIN EmployeeTerritories et ON t.TerritoryID=et.TerritoryID
INNER JOIN Employees e ON et.EmployeeID=e.EmployeeID--
doing multiple Joins so the table can be linked
INNER JOIN Orders o ON e.EmployeeID=o.EmployeeID
INNER JOIN [Order Details] od ON o.OrderID=od.OrderID
GROUP BY t.RegionID
HAVING SUM(od.Quantity*od.UnitPrice*(1-od.Discount)) > 1000000

--1.7
SELECT COUNT(o.OrderID) AS "Freight greater than 100"
FROM Orders o
WHERE o.Freight>100 AND (o.ShipCountry='USA'OR o.ShipCountry='UK')

--1.8
SELECT TOP 1 od.OrderID,SUM(od.UnitPrice*od.Discount*od.Quantity) AS "Highest Value"
FROM [Order Details] od
GROUP BY od.OrderID
ORDER BY "Highest Value" DESC


--------------------------------------------------------------------------------
--QUESTION 2
```

```
--2.1
CREATE DATABASE abdul_db;-- creates a database names abdul_db

CREATE TABLE spartan_table-
- creates a table that has the following attributes with their following types specified
(
    spartan_id INT IDENTITY(1,1) PRIMARY KEY,
    spartan_info VARCHAR(500),
    title VARCHAR(5),
    first_name VARCHAR(30),
    last_name VARCHAR(30),
    university_name VARCHAR(30),
    course_taken VARCHAR(30),
    mark_achieved CHAR(3) NOT NULL
);

--2.2
INSERT INTO spartan_table--specifies which attribute you want to add to
(
    spartan_info,title,first_name,last_name,university_name,course_taken,mark_achieved
)
VALUES--allows you to add values to the attributes specified in the previous block of code
(
    'fjsfdh fddfhj','Mr','dfsfdsf','dfsdf','dfdf','ddwdfe','2.2'

);

SELECT * FROM spartan_table;


-------------------------------------------------------------------------------------
--QUESTION 3

--3.1
SELECT e.FirstName+' '+e.LastName AS "Employee Name",s.FirstName+' '+s.LastName AS "Reports
 to"--
Concatenates the first name and last name, also makes it so you have the employyes name col
umn and who they report to column
FROM Employees e LEFT JOIN Employees s ON e.ReportsTo=s.EmployeeID

--3.2
SELECT s.CompanyName, ROUND(SUM(od.UnitPrice*od.Quantity*(1-od.Discount)),2) AS "Total"--
Finds the total price of a particualr item to 2d.p.
FROM Suppliers s INNER JOIN Products p ON s.SupplierID=p.SupplierID --
joining tables together
INNER JOIN [Order Details] od ON p.ProductID=od.ProductID
GROUP BY s.CompanyName-- groups it by company name so only they are summed
HAVING SUM(od.UnitPrice*od.Quantity*(1-od.Discount)) > 10000 --
limits it so only the companies that made over 10000 are in the table
ORDER BY 'Total' DESC
```




```
--3.3
SELECT TOP 10 c.CompanyName, ROUND(SUM(od.Quantity*od.UnitPrice*(1-
od.Discount)),2) AS "Value" --
top 10 limits the values according to the question and the sum figures out the total value
of the order shipped. Round limits it to 2 decimal places.
FROM Customers c INNER JOIN Orders o ON c.CustomerID = o.CustomerID
INNER JOIN [Order Details] od  ON o.OrderID = od.OrderID
WHERE YEAR(o.OrderDate) = YEAR((SELECT TOP 1 OrderDate FROM Orders ORDER BY OrderDate DESC)
) AND o.ShippedDate IS NOT NULL--
the is not null prevents unshipped orders from being included
GROUP BY c.CompanyName--grouping so the sum taken is only for the same company name
ORDER BY "Value" DESC --Ordering in descending order

--3.4
SELECT FORMAT(o.OrderDate, 'yy-MM') AS "Year-Month",--
Fomatting date to just years and month
AVG(DATEDIFF(D,o.OrderDate, o.ShippedDate)) AS "Average Ship Time" FROM Orders o --
working out the average by finding the difference between shipped date and order date in da
ys and using the average function.
GROUP BY FORMAT(o.OrderDate, 'yy-MM') --
grouping it so only the dates with the same dates are averaged
```