1. ***Sales Representatives in the United States***
Now we'd like to see only for those employees that both have the title of Sales Representative, and also are in the United States

```
SELECT * FROM Sales WHERE title = 'Sales Representative' AND Country = 'United States';
```

2. ***Orders placed by specific EmployeeID***
Show all the orders placed by a specific employee. The EmployeeID for this Employee (Steven Buchanan) is 5.

```
SELECT * FROM Orders WHERE EmployeeID = 5;
```

3. ***Suppliers and ContactTitles***
In the Suppliers table, show the SupplierID, ContactName, and ContactTitle for those Suppliers whose ContactTitle is not Marketing Manager.

```
SELECT SupplierID, ContactName, ContactTitle
FROM Suppliers
WHERE ContactTitle <> 'Marketing Manager';
```

4. ***Products with "queso" in ProductName***
In the products table, we'd like to see the ProductID and ProductName for those products where the ProductName includes the string "queso".

```
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE '%queso%';
```

5. ***Orders shipping to France or Belgium***
Looking at the Orders table, there's a field called ShipCountry. Write a query that shows the OrderID, CustomerID, and ShipCountry for the orders where the ShipCountry is either France or Belgium.

```
SELECT OrderID, CustomerID, ShipCountry
FROM Orders
WHERE ShipCountry = 'France' OR ShipCountry = 'Belgium';
-----------------------------------------------------------------------------------------
SELECT OrderID, CustomerID, ShipCountry
FROM Orders WHERE ShipCountry IN ('France', 'Belgium');
```

6. Orders shipping to any country in Latin America
Now, instead of just wanting to return all the orders from France of Belgium, we want to show all the orders from any Latin American country. But we don't have a list of Latin American countries in a table in the Northwind database. So, we're going to just use this list of Latin American countries that happen to be in the Orders table: Brazil Mexico Argentina Venezuela It doesn't make sense to use multiple or statements anymore, it would get too convoluted. Use the in statement.

```
SELECT OrderID, CustomerID, ShipCountry FROM Orders
WHERE ShipCountry IN ('Brazil', 'Mexico', 'Argentina', 'Venezuela');
```

7. Employees, in order of age
For all the employees in the Employees table, show the FirstName, LastName, Title, and BirthDate. Order the results by BirthDate, so we have the oldest employees first

```
SELECT FirstName, LastName, Title, BirthDate
FROM Employees ORDER BY BirthDate ASC;
```

8. Showing only the Date with a DateTime field
In the output of the query above, showing the Employees in order of BirthDate, we see the time of the BirthDate field, which we don't want. Show only the date portion of the BirthDate field.

```
SELECT FirstName, LastName, Title, DATE(BirthDate) AS BirthDate
FROM Employees ORDER BY BirthDate ASC;
```

9. Employees full name
Show the FirstName and LastName columns from the Employees table, and then create a new column called FullName, showing FirstName and LastName joined together in one column, with a space in-between

```
SELECT FirstName, LastName, CONCAT(FirstName, ' ', LastName) AS FullName
FROM Employees;
```

10. OrderDetails amount per line item
In the OrderDetails table, we have the fields UnitPrice and Quantity. Create a new field, TotalPrice, that multiplies these two together. We'll ignore the Discount field for now. In addition, show the OrderID, ProductID, UnitPrice, and Quantity. Order by OrderID and ProductID.

```
SELECT OrderID, ProductID, UnitPrice, Quantity, UnitPrice * Quantity AS TotalPrice
FROM OrderDetails ORDER BY OrderID, ProductID;
```

11. How many customers?
How many customers do we have in the Customers table? Show one value only, and don't rely on getting the recordcount at the end of a resultset.

```
SELECT COUNT(*) AS TotalCustomers FROM Customers;
```

12. When was the first order? Show the date of the first order ever made in the Orders table

```
SELECT MIN(OrderDate) AS FirstOrderDate FROM Orders;
```

13. Countries where there are customers. Show a list of countries where the Northwind company has customers.

```
SELECT Country FROM Customers GROUP BY Country;
```

14. Categories, and the total products in each category For this problem, we'd like to see the total number of products in each category. Sort the results by the total number of products, in descending order.

```
SELECT CategoryID, COUNT(*) AS TotalProducts FROM Products
GROUP BY CategoryID
ORDER BY TotalProducts DESC;
```

15. Categories, and the total products in each category
For this problem, we'd like to see the total number of products in each category. Sort the results by the total number of products, in descending order.

```
SELECT c.CategoryName, COUNT(p.ProductID) AS TotalProducts
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName ORDER BY TotalProducts DESC;
```

16. Total customers per country/city In the Customers table, show the total number of customers per Country and City

```
SELECT Country, City, COUNT(*) AS Total_Customers
FROM Customers
GROUP BY Country, City;
```

17. ***Products that need reordering***

What products do we have in our inventory that should be reordered? For now, just use the fields UnitsInStock and ReorderLevel, where UnitsInStock is less than the ReorderLevel, ignoring the fields UnitsOnOrder and Discontinued. Order the results by ProductID.

```
SELECT * FROM Products WHERE UnitsInStock < ReorderLevel ORDER BY ProductID;
```

18. ***Products that need reordering, continued***

Now we need to incorporate these fields—UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued—into our calculation. We'll define "products that need reordering" with the following: UnitsInStock plus UnitsOnOrder are less than or equal to ReorderLevel The Discontinued flag is false (0)

```
SELECT * FROM Products
WHERE (UnitsInStock + UnitsOnOrder) <= ReorderLevel AND Discontinued = 0
ORDER BY ProductID;
```

19. ***Customer list by region***

A salesperson for Northwind is going on a business trip to visit customers, and would like to see a list of all customers, sorted by region, alphabetically. However, he wants the customers with no region (null in the Region field) to be at the end, instead of at the top, where you'd normally find the null values. Within the same region, companies should be sorted by CustomerID

```
SELECT CustomerID, CompanyName, Region
FROM Customers
ORDER BY Region IS NULL, Region ASC, CustomerID ASC;
-----------------------------------------------------------------------------------------------------------
SELECT CustomerID, CompanyName, Region,
    CASE WHEN Region IS NULL THEN 1 ELSE 0 END AS RegionSort
FROM Customers ORDER BY RegionSort, Region ASC, CustomerID ASC;
```

20. ***High freight charges***

Some of the countries we ship to have very high freight charges. We'd like to investigate some more shipping options for our customers, to be able to offer them lower freight charges. Return the three ship countries with the highest average freight overall, in descending order by average freight

```
SELECT ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
LIMIT 3;
```

***21.** High freight charges - 2015*
We're continuing on the question above on high freight charges. Now, instead of using all the orders we have, we only want to see orders from the year 2015.

```
SELECT ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
WHERE YEAR(OrderDate) = 2015
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
LIMIT 3;
```

***22.** High freight charges - last year*
We're continuing to work on high freight charges. We now want to get the three ship countries with the highest average freight charges. But instead of filtering for a particular year, we want to use the last 12 months of order data, using as the end date the last OrderDate in Orders.

```
SELECT ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
WHERE OrderDate >= (
   SELECT DATE_SUB(MAX(OrderDate), INTERVAL 12 MONTH)
   FROM Orders)
GROUP BY ShipCountry
ORDER BY AverageFreight DESC
LIMIT 3;
```

***23.** Customers with no orders*
There are some customers who have never actually placed an order. Show these customers

```
SELECT c.CustomerID, c.CompanyName, FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderID IS NULL;
```