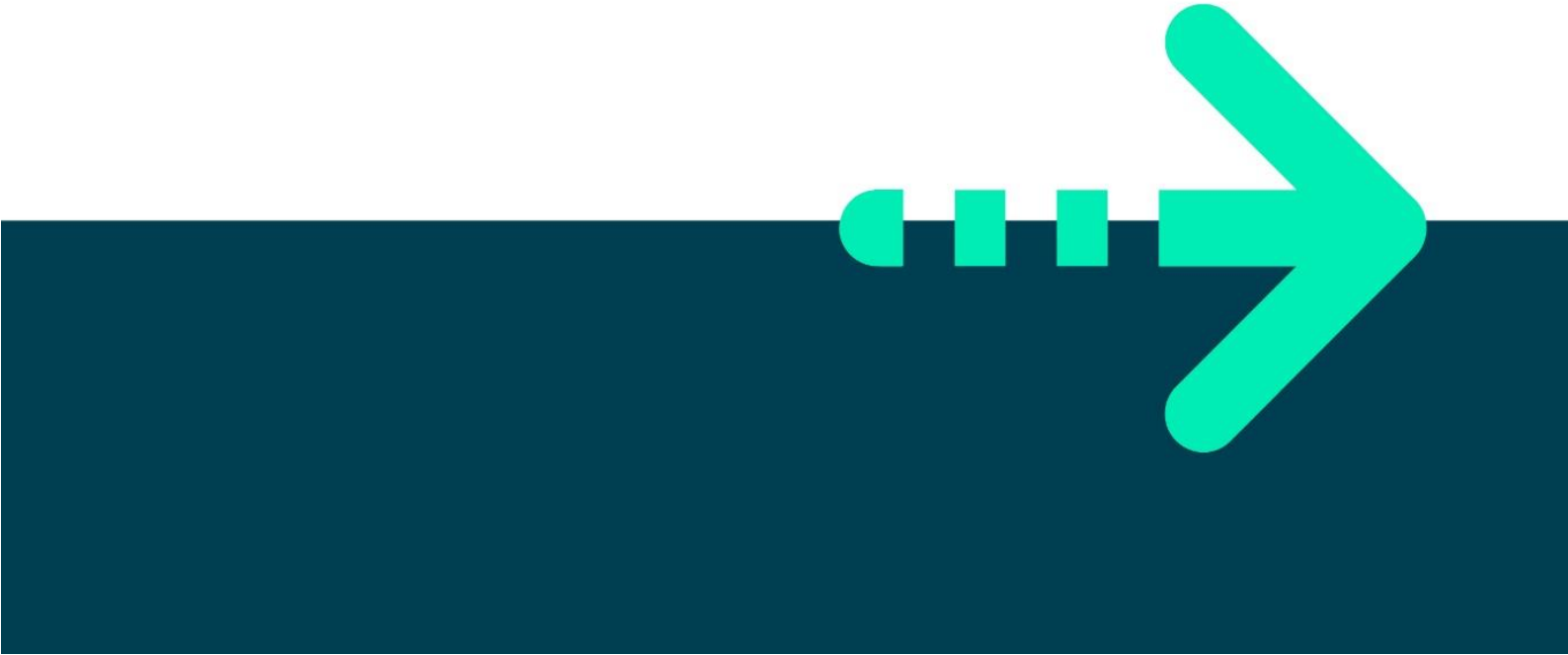
Lab: Using Multiple Tables

Data Analyst Level 4



Lab: Using Multiple Tables

Exercise 1: Join fundamentals

You need to write a report for Northwind's UK-based customers and the orders that they have placed.

In this exercise, you will use inner joins, the fundamental type of join in SQL.

The main tasks for this exercise are as follows:

1. Create a report that selects the customer ID, city and company and contact names from the Customers table and the order number and order date from the Orders table.
2. Modify the report so that it only picks up orders placed by UK Customers and sorts the results by customer and order date.
3. Modify the report so that it also includes details about the name of the product that was ordered, and how many units of each product were ordered.

Task 1: Create a report that selects rows from the Customers and Orders tables

1. Create a new query and save it with a name of "UKCustomerOrders.sql".
2. Write a report that uses the Northwind database and selects the CustomerID, CompanyName, ContactName and City columns from the dbo.Customers table. Alias the table as 'c'.
3. Execute the query and verify that 91 rows are returned.
4. Modify the query to join rows from the dbo.Orders table that have the same value in the CustomerID column. Alias Orders as 'o'.
5. Add the OrderID and OrderDate columns from the Orders table to the select list.
6. Execute the query and verify that 830 rows are returned.

Task 2: Write a query that filters and sorts the results

1. Modify your existing report.
2. Add a where clause to limit the rows returned to those Customers whose Country column is equal to UK.
3. Execute the query and verify that 56 rows are returned, all of whose City columns are British cities.

4. Add an order by clause to ensure that results are sorted on CompanyName then OrderDate.
5. Execute the query and verify that 56 rows are returned, the first one is for AROUT from November 1996 and the last one is for SEVES from February 1998.

Task 3: Write a query that includes rows from more than two tables

1. Make a copy of your existing query.
2. Join the dbo.[Order Details] table aliased as 'od'. Use the OrderID column in Orders and Order Details to find matching rows.
3. Add the ProductID and Quantity columns to the select list. You can also remove the City column from the earlier queries – that was just to add a visual verification that you were only getting UK customers.
4. Execute the query and verify that 135 rows are returned – we're getting more rows back because we're selecting every row down at the Order Detail level now.
5. Join yet another table – the Products table. Alias it as 'p' and use the ProductID column from Order Details to find the matching rows. Include the ProductID and ProductName columns in your select list.
6. Execute the query and verify that the first order is for 25 lots of Guarana Fantastica and the last one is for 20 Scottish Longbreads.
7. [Extra Credit] limit the results to only include products in the Seafood category and display the category name as well (18 rows)

Exercise 2: Investigate outer joins

In this exercise you will investigate left and right outer joins. You will also revisit aggregation from an earlier exercise.

The main tasks for this exercise are as follows:

1. Create a query that gets a count of Customers.
2. Create a query that selects the Company Name from Customers and the number of orders for those customers from the Orders table.
3. Modify the report to retrieve all customers, regardless of whether they have placed orders and add a column showing the minimum order date to the select list.

Task 1: Create a query that counts customers

1. Create a new query and save it with a name of "CustomerOrderAnalysis.sql".
2. Write a report that uses the Northwind database and displays a count of all the rows in the dbo.Customers table.
3. Execute the query and make a note of the result.

Task 2: Write a report that groups orders based on the customer's name

[NOTE: This is similar to a task in an earlier lab.]

1. Comment out or delete the Customer count query
2. Write a new query that selects the CompanyName column from the Customers table and a count of OrderID columns from the Orders. Alias the count column as 'NumOrders'. You will need a group by clause.
3. Add an order by clause to sort the records in number of orders placed.
4. Execute the query and make a note of the number of rows returned. You will see that it differs from the number of customers. Take a few moments to think about why that might be.
   Hint: What is the lowest count of orders?

Task 3: Write a report that uses left and right outer joins

1. Modify the existing report.
2. Change the query so that it selects all of the rows from the Customers table and those rows from the Orders table where there is a match.

3. Execute the query and verify that it now returns 91 rows, the top two now being FISSA and Paris Specialités, both with a count of 0.
4. Change the left outer join to a right outer join and confirm that you're back to only seeing 89 rows. In this case, we get the same results as for an inner join.
5. Change the order in which the tables are listed so that the right outer join returns 91 rows.
6. Finally, add another column to the query that selects the minimum order date from the Orders table and alias it as 'MinDate'.
7. Execute the query and note that we have a couple of null values for the inactive customers The COUNT aggregate was able to come up with a total of zero for those customers but the MIN has no values to work on so it must return null. In an actual report, you might want to tidy that up so that it shows some text instead.

Exercise 3: Create a telephone directory

Northwind Traders want to create a centralised telephone directory for all suppliers, customers, and employees.

In this exercise, you will use the UNION operator to combine results.

The main tasks for this exercise are as follows:

1. Create a report that retrieves the company name, contact name and telephone number of all customers.
2. In the same query, create a report that retrieves the same columns for all suppliers.
3. In the same query, create a report that retrieves the full name and extension number for all employees.
4. Combine the results using a UNION operator.

Task 1: Create a report that retrieves Customer contact details

1. Create a new query and save it with a name of "ContactDirectory.sql".
2. Write a report that uses the Northwind database and selects the CompanyName, ContactName and Phone columns from the dbo.Customers table.
3. Execute the query and verify that it returns 91 rows.

Task 2: Create a report that retrieves Supplier contact details

1. In the same script file, add another query that selects the same three columns from the Suppliers table.
2. Select only that part of the script and execute the query. Verify that 29 rows are returned.

Task 3: Create a report that retrieves Employee contact details

1. In the same script file, add another query that selects rows from the Employees table.
2. In the select list for the query, add a column that concatenates the FirstName column to a space character and the LastName column (building up a string containing the employee's full name).
3. Add the Extension column as well.
4. Execute the query and verify that you retrieve 9 rows of two columns – one containing the full name and one containing the extension.

Task 4: Combine the results using the UNION operator

1. In the same script file, between the customers query and the suppliers query, add the UNION keyword.
2. Select both parts of the script and execute. Verify that you get 120 rows back. Scroll through the results and note that they are sorted alphabetically by company name.
3. Now add the UNION keyword between the suppliers query and the employees query and attempt to run the whole script.
4. You get an error, because the employees query only has two columns in it whilst the other two have three columns.
5. You need to add a company name column to the employee's part of the query. Now, the Employees table doesn't have a company name column but that's not a problem because we actually know what company all of the employees work for – Northwind!
6. At the start of the select list for employees, add the string 'Northwind Traders'.
7. Run the whole query again and verify that 129 rows are returned. Scroll through the list and note that the Northwind Traders employees are all floating around in the middle of the results.
8. Change the UNIONs to UNION ALL and rerun the query. This time all the employees are right at the very end. Also, although you probably won't have noticed, the query has run considerably quicker than before as it hasn't tried to remove duplicates by sorting first.
9. Feel free to add an order by clause if you want to!