Day 4

Intermediate SQL

1. Subqueries

Ex 1:

Create a query which lists out all of the events in the **tblEvent**table which happened after the last one for country 21 (**International**) took place.

Ex 2:

Write a sub query to filter events to show only those which have an event name of longer than average length. You will need the **AVG** and **LEN** functions to do this.

*Hint: You can run your subquery separately by highlighting it to show that the average****EventName****length is 26 characters*

Ex 3:

Write a **SELECT** statement to return events from the 3 continents with the fewest events.

*Hint: To do this first write a SELECT query which returns all the continents and events.*

*Now underneath write another****SELECT****statemnt which lists  events for the 3 continents with the lowest****COUNT****of events. Put the****COUNT****in the****ORDER BY****clause, not the****SELECT****.*

*Finally use the second****SELECT****as a filter in the first****SELECT****'s****WHERE****clause. To do this use****ContinentName******IN****(Sub Query).*

Ex 4:

Write a query which lists out countries which have more than 8 events, using a correlated subquery rather than **HAVING**.

Ex 5:

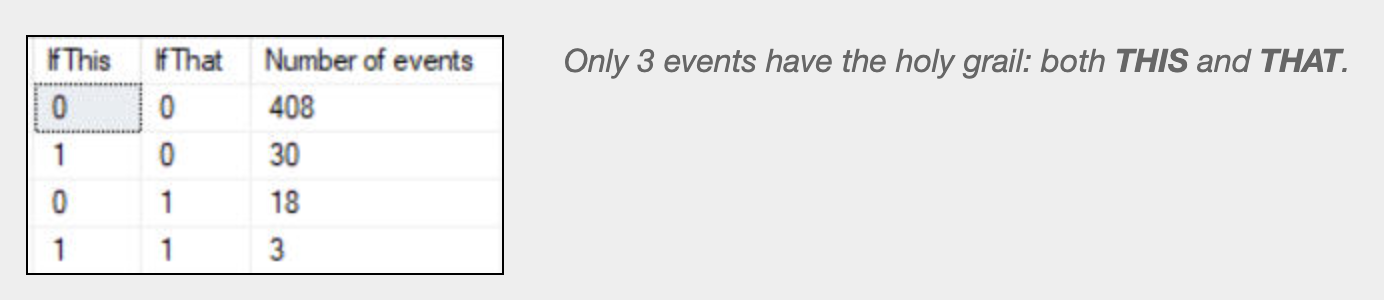
Create a subquery to list out all of those events whose:

* Country id is not in the list of the last 30 country ids from tblCountry in alphabetical order; and
* Category id is not in the list of the last 15 category ids from tblCategory in alphabetical order.

1. CTEs

Ex 1:

The aim of this exercise is to show the number of events whose descriptions contain the words this and/or that:



In a single query solution you would have to use a **CASE** expression to determine the value of **IfThis** and **IfThat**, then group by the same **CASE** expression.  This is messy, and makes it hard to make subsequent changes to your expression (as you have to do it in two places).

Create a query to solve this problem in two passes:

Graphical user interface, application

Description automatically generated

If you get this working and still have spare time, try changing or extending your query to show the 3 events whose details contain both this and that:

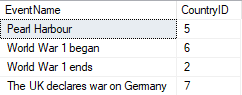
Graphical user interface, text, application

Description automatically generated

Other than the text they contain, there's no obvious link between these 3 events.

Ex 2:

The aim of this exercise is to use a derived table to hold data before joining this another table. Start by selecting only events which end with a letter between **N** and **Z**.



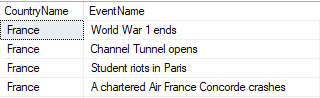
*Your query should return 300 results (including the****CountryID****column will allow us to join to the****tblCountry****table).*

Now turn this select into a derived table called **Second\_Half\_Derived.**The layout will be something like this:

Graphical user interface, text, application

Description automatically generated

Now join **tblCountry** to the derived table as you would for a normal table:



*This should return 230 results - scroll down for some happier events.*

Ex 3:

The aim of this exercise is to use a CTE (Common Table Expression) to extract data from the database in two passes:

1. Get a list of all of the episodes written by authors with **MP**in their names.
2. Get a list of the companions featuring in these episodes.

Start by creating a query to show just the episode id numbers for those Doctor Who episodes written by authors with MP as part of their names:

Text

Description automatically generated

Your final query should join to your CTE to show the companions featuring in these episodes (you may need to use the word DISTINCT to avoid some companions' names appearing twice):

Graphical user interface, text

Description automatically generated

Ex 4:

The aim of this exercise is to use CTEs to answer the following questions:

Graphical user interface, text, application, email

Description automatically generated

\*Note: Filter on EventDetails for Step 1.

If you display the results in date order, you should get this:

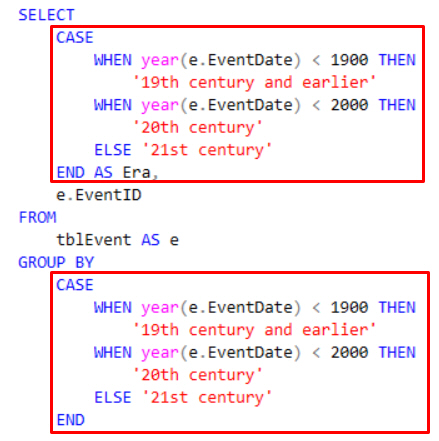


*The first few of the 116 events returned by your final query.*

Save your query as **Owl-free events**, then close it down.

Ex 5:

The aim of this exercise is to count the number of events by a column called **Era** which you'll calculate, without including this calculation twice:



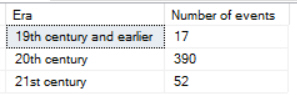
*Here's what NOT to do - we want to avoid repeating the same calculation twice.*

To do this, you can do the calculation in two bites, using a CTE to hold the intermediate stage.  First create a query to show the era for each event:



*You could peek at the expression at the start of this exercise to get help on how to calculate the era!*

Now store this as a CTE, and write a query using it which shows the number of events per era:



*What the entire query should show when you run it.*

Save this query as **Epoch-making**, and close it down.