Sub Queries

- Overview
- Append Queries
  - Example
- Union Queries
  - Examples
  - Union All
- Intersect Queries
  - Example
- Minus Queries
  - Example
- Advanced Sub Queries
  - Example

Overview

top

Sub queries permit as user to generate far more sophisticated reports. For example if you wanted to compare the sales results of this financial year with past years you may wish to use an append query or if you wanted to determine which customers were new in a particular year you would use a minus query. In both these examples Yellowfin is generating two distinct queries and then combining the result set to provide you with a single table of results. With Yellowfin you can create 4 types of sub query, these are:

	**Append**	The append sub query takes the results of one query and appends them to another set of results as additional columns. The purpose of this query is to allow the result of one query to be compared to that of another.
	**Union**	The union sub query combines the results of two queries into a single table of matching columns. Unions can be useful in a data warehouse application when tables aren't perfectly normalised.
	**Minus**	The minus sub query takes distinct rows of one query and returns the rows that don't appear in the second query.
	**Intersect**	The intersect query takes the results of two queries and returns only the rows that appear in both sets.

Append Queries

top

The append sub query takes the results of one query and appends these to another as new columns of data. The two queries must have exactly the same GROUP BY (or Dimensional) columns in order to join them.

The purpose of the append query is to allow the result of one query to be compared with another. For example you may want to compare the YTD revenue for the current year with the revenue for the same period for last year. Typically this is difficult to do unless the data source has been configured to allow this by having a column for each of these attributes. Generally though the data will be stored a separate rows in the same table. With the append query, 1 query will retrieve the current period's results whilst the other will retrieve the previous periods result. Using calculated fields it is now possible to compare the results.

Common Column	Metric 1
Description A	1234
Description B	1152

Filter 1

+

Common Column	Metric 2
Description A	5586
Description B	9876

Filter 2

=

Common Column	Metric 1	Metric 2
Description A	1234	5586
Description B	1152	9876

Filter 1 Filter 2

Example

Comparing in revenue in one year compared to another by country.

1. Firstly you will need to create a query that returns the revenue for a selected period by country. Country, Sum Invoiced Amount and Year in the filter
2. Now select the sub query option and choose append. You will see a very similar query builder to the standard builder. In the fields section you will have to replicate the attributes of your original query so that the same level of aggregation can occur.
3. The join will have to be specified for the sub query. In the join section click the refresh link to display the available join fields. You will have to link the fields on the master query with the fields on the sub query.
   Note: you do not have to include metric fields.
4. Once you have matched the fields click the add icon to add the join to the list.

5. Now return to your master query. You should see additional attributes in the fields list. Note they are prefixed by sub query: and cannot be removed from the list of fields.
6. The final step is to set the filters. Progress to the filters page. You will see that similar to the data page you have a tabbed set of filter attributes. Set the filter value for each filter for the specific periods required.
7. When you now run the report you will have two columns one for each period specified. If required you can also created calculated fields to determine the difference between the two values. This would be done on the master query by creating calculated fields in the standard way.

> *See Append Sub Query Tutorial for more information.*

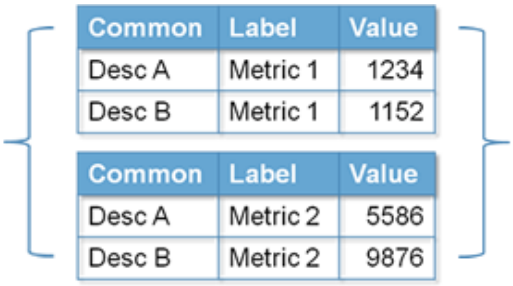
Union Queries

top

A Union query combines the results of two SQL queries into a single table of all matching rows. The two queries must have the same number of columns and compatible data types in order to join them. Any duplicate records are automatically removed unless UNION ALL is used.

Union can be useful in data warehouse applications where tables aren't perfectly normalised. For example a table may have revenue in separate columns for individual products rather than revenue by product code. A union join would allow reporting to change display from:

Common	Metric 1	Metric 2
Desc A	1234	5586
Desc B	1152	9876



Common	Label	Value
Desc A	Metric 1	1234
Desc B	Metric 1	1152

Common	Label	Value
Desc A	Metric 2	5586
Desc B	Metric 2	9876

Common	Label	Value
Desc A	Metric 1	1234
Desc A	Metric 2	5586
Desc B	Metric 1	1152
Desc B	Metric 2	9876

Examples

Display invoiced amount and cost amount on separate lines rather than separate columns.

1. Create a calculated field for a label = Invoiced
2. Create a query that returns the revenue by country. Country, Label and Sum Invoiced Amount
3. Now select the sub query option and choose Union. With the union query you will need to match fields from the first query with those in the second. In this case a new calculated field is created for the “Cost Label” and added into the report.
   Instead of the metric invoiced amount the new metric of cost of camp is added.
4. When you now run the report you will have a single column for both received and invoiced amounts.

> *See Union Sub Query Tutorial for more information.*

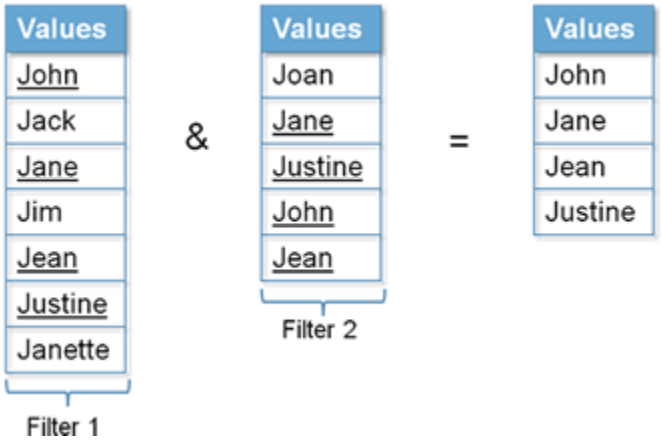
Union All

With a standard union join duplicate records are not returned so if the row in the sub query matches the row in the master query it will not be displayed, If you wish to display duplicate records you must specify this at the sub query level.

Intersect Queries

top

An Intersect query takes the results of two queries and returns only rows that appear in both result sets. For example if you wanted to know which customers purchased services in Year 1 **as well as** Year 2 then an intersect query is needed.



Example

Determine which customers purchased services in Year 1 **as well as** Year 2.

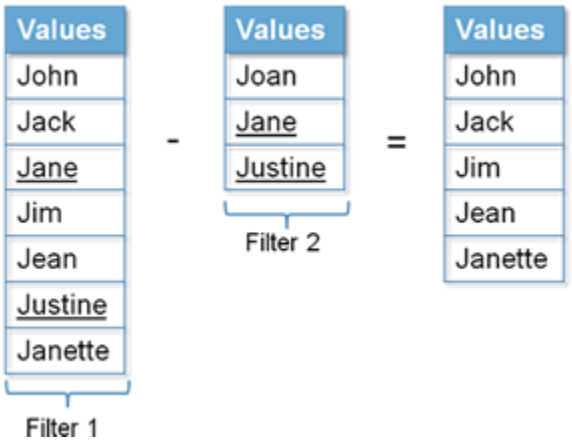
1. Firstly you will need to create a query that returns a list of customers that took part in Year 2. Athlete ID, First Name, Last Name and Year in the filter.
   Note: some of these may not have purchased in Year 1.
2. Now select the sub query option and choose Intersect. You will see a slightly different interface to the normal query builder. The purpose of this is to select a linked field or key in the master query and determine which filters you which to apply. In this example we want to link on the athlete id and filter it by year 1.
3. The final step is to set the filters. Progress to the filters page. You will see that similar to the data page you have a tabbed set of filter attributes. Set the filter value for each filter for the specific periods required.

> *See Intersect Sub Query Tutorial for more information.*

Minus Queries

top

A Minus query takes the distinct rows of one query and returns the rows that do not appear in a second result set. A minus query is almost the opposite of the intersect query, rather than displaying data in common the minus subtracts data from the result set.



Example

Determine which customers purchased services in Year 1 and **never before**.

1. Firstly you will need to create a query that returns a list of customers that took part in Year 1. Athlete ID, First Name, Last Name and Year in the filter

2. Now select the sub query option and choose Minus. You will see a slightly different interface to the normal query builder. The purpose of this is to select a linked field or key in the master query and determine which filters you which to apply. In this example we want to link on the athlete id and filter it **different from** year 1.
3. The final step is to set the filters. Progress to the filters page. You will see that similar to the data page you have a tabbed set of filter attributes. Set the filter value for each filter for the specific periods required. Master query = Year of interest, Sub Query **Different From** Year of Interest.

> *See Minus Sub Query Tutorial for more information.*

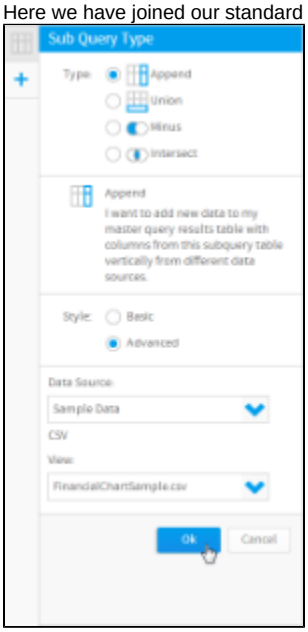
Advanced Sub Queries

top

Advanced Sub Queries extend the basic Sub Query functionality, by allowing you to query multiple views and, in some cases, data sources.

Example

Here we have joined our standard Tutorial connection and Ski Team view content with a CSV imported into a different database.



> *See Advanced Sub Query Tutorial for more information.*

top