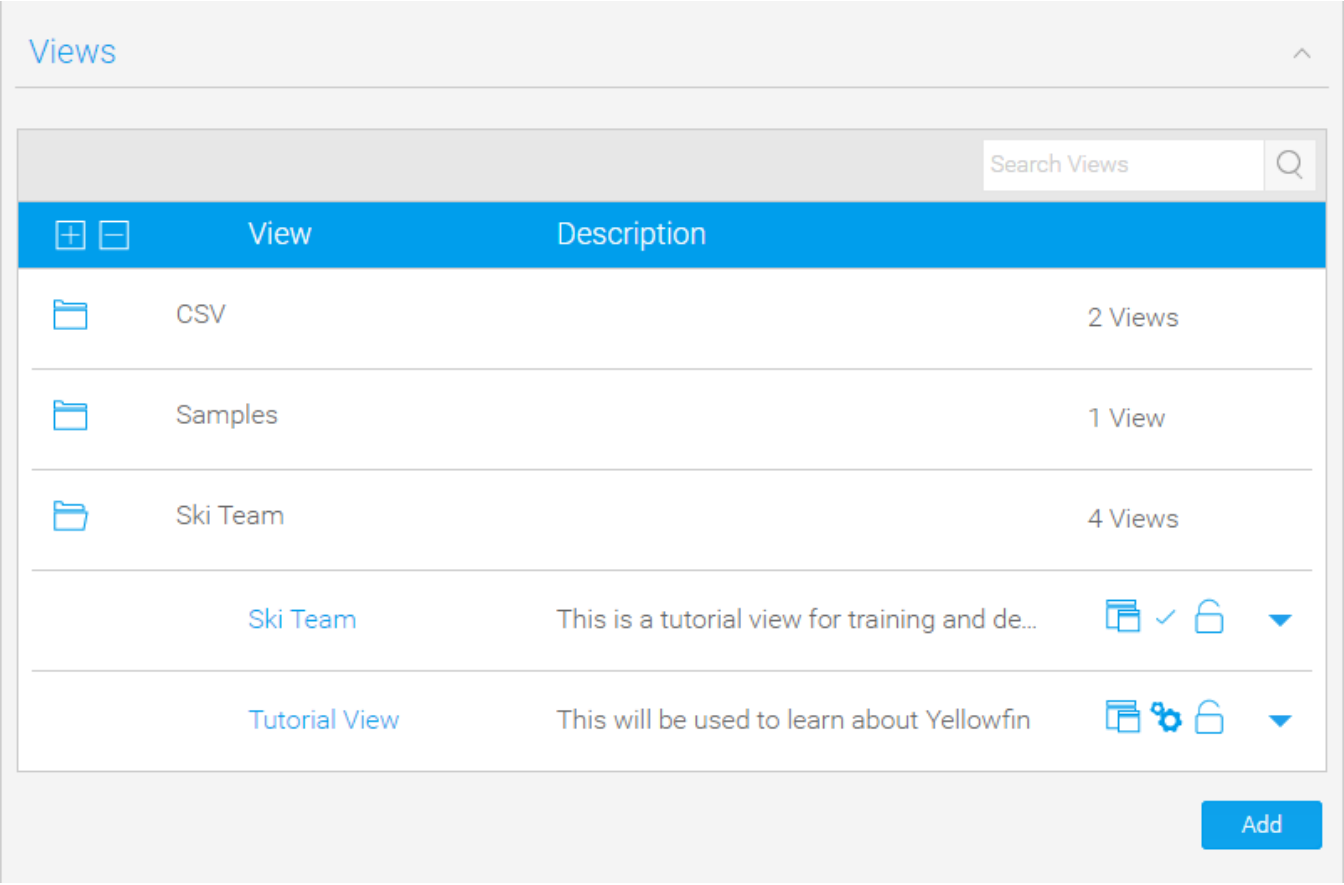
Views

Overview

top
A Yellowfin View is a metadata layer that sits between the Source Connection and the Report Builder. It's used to define relationships between tables, identify fields to be accessed by report writers, and define default formatting for these fields. A report writer will use the relationships and fields defined in the view to base their reports on, without having to understand the underlying logic.



> *See View Creation for more information.*

Who is the view Administrator?

Views are created by an administrator. There is no standard profile for a view administrator. Within a company, the person designated as the view administrator may be the database administrator, an applications manager or developer, a project manager, or a business user who has acquired enough technical skills to create views for other users.

A view administrator should have the following skills and level of technical knowledge:

Skill /Knowledge	Description

Ability to analyse user needs	The view administrator must have the skills to conduct user needs analyses to create categories and fields that are relevant to the user vocabulary, and to develop views that meet the needs of the user community.
Database knowledge	A View administrator needs to have a good working knowledge of the company's database management system (DBMS), how the databases are deployed, the logical database structure, and the type of data stored in company databases.
Structured Query Language (SQL)	A working knowledge of SQL is necessary.

View Components

top
A view contains the following structures:

Categories	The purpose of field categories is to provide logical groupings of fields within a view. The name of a category should intuitive to the business user and provide an indication of the fields that it is likely to contain. For example a category called 'Private details' is likely to contain a person's name, age and gender.
Fields	A field is a named component that maps to data or a derivation of data in the database. The name of a field should be drawn from the business vocabulary of the targeted user group. For example, fields used in a view used by a product manager could be Product, Life Cycle, or Release Date. A view used by a financial analyst could contain fields such as Profit Margin, and Return on Investment. The fields that report writers see in a view infer SQL structures that have been inserted into a database schema.

Field Types

When creating a VIEW, you define and categorise fields. The definition of a field reveals how it can be used in analysis and reports. A field can be defined as a dimension or a metric. Each type of field serves a different purpose:

	Dimension fields retrieve the data that will provide the basis for analysis in a report. Dimensions typically retrieve character-type data (employee names, company names, etc.), or dates (years, quarters, etc.)
	Metric fields retrieve numeric data that is the result of calculations on data in the database. Metrics tend to be dynamic: the values they return depend on the dimensions they are used with. For example, if you include Person and Age in a query, Age per person is calculated.
	View Geography Fields are fields linked to a GeoPack.
	Pre-Defined Filters are fields where a set of conditions have been set up when the view was created. This assists users to limit the data returned in a query to only the expected results. For example if the filter is called 'United States' then only data from the united states would be included in the results.
	Parameters are fields which are used to capture user defined values and pass them into calculated fields or filters. These parameters can assist in conducting what if analysis.
	View Filter Groups are sets fields to be used as filters, reused multiple times. Filter Groups can contain filter dependency hierarchies, as well as cached values. These only have to be set up once, rather than for each report.

View Use

top
Views are used by Yellowfin report writers. The view meta-data is stored within the Centralised Yellowfin repository. An end user connects to a view via a web browser when running a report.

By using a view, the end user automatically has access to data within your source system. Access to data is restricted by the fields that are available in the view. These fields have been created by the administrator based on the report users needs.

Assisting Data Analysis

A view can represent the data needs of any specific application, system, or group of users. For example, a view can contain fields that represent the data needs of the Marketing or Accounting departments in a company.
A view can also represent the data needs of a section within a department or any set of organized procedures such as a payroll or inventory system.

Who uses views?

Yellowfin report writers use views for reporting and analysis. The view should provide them with categories and fields relevant to their business domain.

View Design

top

The view design methodology consists of four stages:

1. Analysis of business problem and planning the view solution
2. Building the view
3. Defining fields and Creating Calculated Fields
4. Publishing the view to users

Each implementation phase is based on an assumption that you have completed an initial planning phase.

1. Plan the view
   The analysis of user requirements and design are the most important stages in the process. Users must be heavily involved in the development process if the view is going to fulfil their needs both with the business language used to name fields and the data that can be accessed.
   Implementation will be very quick and easy if this stage is carried out properly. You should note the following points:
   a. You must fully understand the data analysis and reporting needs of the target audience for the view. Do not create fields by looking at the columns available in the database, but identify columns that are required as a result your user needs analysis.
   b. Understand the source system data and business rules required for generating the required fields for users.
2. Building the view
   You create an entity relationship diagram for the underlying database structure of your view. This includes the selecting the appropriate tables and columns of the source database and the joins by which they are linked.
3. Defining Fields
   Select columns form your source system tables and organise these fields into categories. These are fields that you have identified from an analysis of user reporting needs. You can create additional calculated fields and filters to enhance user reporting capabilities and optimise query performance.
   Test the integrity of your view structure. You should also perform tests using the report writer on the view.
4. Publish the View
   You can publish your view to users for testing, and eventually for production use, by expanding the number of users that have access to the view.

The table below outlines the major phases in a typical view development cycle:

Development phase	Description
Prepare	Identify the target data source and become familiar with its structure. Know what data is contained within each table required for the view and the joins that related the tables to each other.
Analyse	Identify what information the users need. Identify what standard reports they require. Familiarise yourself with their business terminology so that you can name fields sensibly. Plan Identify a project strategy. For example, how many views should be created and which ones should have the capacity to be linked and to what level.
Implement	Build the view either on the database or through the Yellowfin view builder. Test frequently during the build process for validity and reliability of inferred SQL.
Test	Form a small group of users, preferably power users who have some knowledge of what information they expect to get from the view. Pre-Release the view to these users by adding them the access security list for the view. Ask the users to perform thorough tests simulating live usage of the view(s).
Deploy	Migrate the view from your Test to Production environments. Change access security of the view so that it is available to the target user base.
Evolve	Update and maintain the view as the data sources and user requirements change and grow.

Note: View design should always be driven primarily by user requirements and NOT the data source structure.

top