REPORT COMPOSER — UX CASE STUDY

Competitive Analysis

Simplifying Report Creation for Nontechnical Users

Rossana · August 2026

Purpose

The goal of this competitive analysis was not to identify the most powerful reporting application. Traditional report-writing tools are designed around sophisticated data configuration and often assume that the person creating the report understands the underlying reporting model.

For the Report Writer redesign, the more relevant question was:

> ***How do applications make complex reporting capabilities usable for people who are not reporting or database specialists?***

The analysis therefore focused on products that allow business users to select, organize, filter, summarize, and present structured data while reducing their exposure to the technical complexity underneath.

Products Evaluated

I evaluated five products representing different approaches to self-service reporting:

Product	Why it was included
Airtable Interfaces	Strong example of presenting complex relational data through a simplified user-facing layer
Smartsheet Reports	Business-oriented reporting built around familiar rows, columns, filters, and groups
Salesforce Report Builder	Mature enterprise reporting designed for business users working with structured application data
HubSpot Custom Report Builder	Guided report creation with a clear progression from data selection to visualization
Zoho Analytics	Useful bridge between approachable self-service reporting and more sophisticated analytical capabilities

The products were not evaluated on feature quantity. The focus was **how effectively each product manages complexity for the user.**

Evaluation Framework

The analysis followed the basic report-building journey:

```
START → SELECT DATA → SHAPE REPORT → REFINE → PREVIEW / OUTPUT
```

Across that journey, I evaluated eight UX characteristics.

Area	Question
Getting Started	Is it clear how to begin without prior training?
Data Selection	Can users choose recognizable information without understanding the database structure?
Field Selection	Is adding information to the report straightforward?
Filtering	Can users describe which records they want using understandable controls?
Grouping & Summaries	Can users organize and calculate data without writing formulas?
Report Structure	Is the relationship between the configuration and resulting report visible?
Preview & Feedback	Can users see the consequences of their choices while building?
Progressive Complexity	Are advanced capabilities available without making basic reporting difficult?

Comparative Findings

01 — Airtable Interfaces

Strongest pattern: ***Hide the data model behind a task-specific interface***

Airtable was particularly relevant because Interface Designer explicitly separates the underlying relational data from what an end user needs to see. An interface creator determines which fields, records, filters, sorting and grouping capabilities should be exposed rather than simply presenting the entire underlying data structure. Airtable describes this as a way to reduce an otherwise overwhelming data layer into smaller, manageable portions for specific users or workgroups.

What works

- Users interact with selected fields rather than the complete data model.
- Search, filter, sort and group can be exposed only when useful.
- Preconfigured layouts establish sensible defaults.
- Record details can be progressively revealed.
- Interfaces can combine information from multiple tables without requiring the end user to navigate those tables directly.

> **Lesson for Report Writer**
> *The underlying data structure does not need to become the user's interface. The system can understand relationships between data while presenting users with recognizable report information.*

02 — Smartsheet Reports

Strongest pattern: ***Build on familiar concepts***

Smartsheet's value as a benchmark is conceptual simplicity. Reporting is framed around familiar spreadsheet and business concepts — rows, columns, criteria and groups — rather than traditional report-development terminology. This represents an important direction for Report Writer: reuse interaction concepts users already understand instead of teaching them the architecture of a reporting engine.

> **Lesson for Report Writer**
> *Reporting should feel like selecting and organizing information, not programming a report.*

03 — Salesforce Report Builder

Strongest pattern: ***Make sophisticated operations incremental***

Salesforce demonstrates how substantial reporting capability can be exposed through individual actions. Users can add fields, filter records, group information and summarize numeric columns without needing to construct those operations as code. For example, numeric columns can be summarized through explicit Sum, Average, Max and Min actions. Filtering similarly starts with understandable field/value conditions before exposing more sophisticated filter logic. Salesforce also changes report structure in response to what the user does. In Lightning Report Builder, for example, the report format updates automatically as data is grouped rather than requiring the user to choose the technical report format first.

What works

- Basic operations remain simple.
- Advanced capabilities are available when needed.
- Summarization is attached directly to the field being summarized.
- Report structure responds to user actions.
- Preview can show sample records while the report is being edited.

> **Lesson for Report Writer**
> *Complexity should emerge from the report the user is building — not be presented before they need it.*

04 — HubSpot Custom Report Builder

Strongest pattern: ***Turn report creation into a sequence***

HubSpot provides perhaps the clearest overall workflow. Its report-building process is explicitly organized as: Select data sources → Add fields → Apply filters → Configure visualization → Save/export. That sequence matters because users aren't asked to understand the entire reporting environment simultaneously. HubSpot also provides search and filtering when choosing fields, and its Smart Chart capability can recommend a visualization based on the fields the user has selected.

What works

- Clear beginning and end.
- Data selection precedes presentation decisions.
- Field search reduces browsing.
- Configuration is contextual to the current stage.
- The system can recommend appropriate presentation choices.
- Advanced AND/OR/NOT filtering exists without becoming the starting experience.

> **Lesson for Report Writer**
> *A report builder can be powerful without presenting all of its power at once.*

05 — Zoho Analytics

Strongest pattern: ***Direct manipulation***

Zoho uses drag-and-drop extensively. Users construct tabular reports by moving fields into the report rather than configuring each field through separate dialogs. Filtering follows the same model: users drag a field into the filter area and are then shown options appropriate to that field's data type.

> *Choose something → then configure that thing.*
> *rather than*
> *Configure the report → configure the data → configure the fields → discover the result.*

> **Lesson for Report Writer**
> *Configuration should follow the object the user is working with.*

Pattern Comparison

UX Pattern	Airtable	Smartsheet	Salesforce	HubSpot	Zoho
Guided starting point	●	●	●	●●	●
Familiar business terminology	●●	●●	●	●●	●
Simplified data selection	●●	●●	●	●●	●
Direct field manipulation	●	●	●	●●	●●
Simple filtering	●●	●●	●	●●	●
Easy grouping / summarizing	●	●●	●●	●	●●
Immediate feedback / preview	●●	●	●●	●●	●●
Progressive complexity	●●	●●	●●	●●	●
Advanced reporting available	●	●	●●	●●	●●

●● Strong pattern ● Present

The point isn't that one product "wins." Each provides a useful model for a different part of the experience.

Key Findings

1. Start with the user's question, not the database

Traditional reporting tools often begin by asking users to understand data sources, tables, relationships or report types. The stronger self-service experiences progressively translate that complexity into recognizable business concepts.

> ***What do you want to report on?***

2. Separate selecting data from configuring it

HubSpot's sequence is particularly useful here: choose the data, add the fields, and then refine how they behave. That suggests SELECT → ORGANIZE → REFINE rather than presenting every field property, filter, format and calculation simultaneously.

3. Let the system manage relationships

Airtable demonstrates the value of creating an interface layer between users and relational data. Users can work with selected information from multiple tables without needing access to everything underneath. For Report Writer, known relationships should therefore be handled by the system whenever possible. Exceptions can be surfaced when the system cannot safely determine the relationship.

> ***Users select information. The application determines how that information is related.***

4. Make the report itself the primary workspace

Zoho's drag-and-drop model establishes a direct relationship between the field being selected and its role in the resulting report. This is substantially easier to understand than configuring an abstract report definition and waiting to see what it produces. The report should increasingly become visible while it is being constructed.

5. Use progressive disclosure for advanced capabilities

Salesforce is especially instructive here. A user can simply summarize a numeric field, but more sophisticated filtering and logic remain available when required. The redesigned Report Writer should follow the same principle: common action first, advanced configuration second. Simple reports should remain simple.

6. Use sensible defaults

The strongest products make decisions for users when enough information exists to do so safely. Airtable can preconfigure filtering and search behavior in its table layouts, while HubSpot can recommend a chart based on selected fields. Report Writer should therefore avoid asking users to make decisions the application can make reliably itself.

7. Keep the result visible

A major usability advantage of contemporary report builders is reducing the distance between configuration and consequence. Salesforce can display sample records as edits are made, while HubSpot and Zoho organize report construction around the report being produced.

> ***“What did that change do?”***

Design Direction

The competitive analysis points toward a different mental model for Report Writer.

Traditional Report Writer

```
DATA MODEL → CONFIGURE → CONFIGURE → CONFIGURE → RUN → DISCOVER RESULT
```

The user has to understand how the report works before they can see whether they built it correctly.

Proposed Report Writer

```
CHOOSE → BUILD → REFINE → PREVIEW → SAVE
```

The system carries more of the technical burden while the user works with concepts they recognize.

This doesn't mean removing sophisticated reporting capabilities. It means **changing when and how users encounter them.**

Design Principles

The analysis produced six principles to guide the redesign:

1. **Speak the user's language**
Present recognizable business information rather than reporting-engine or database terminology.

2. **Select before configuring**
Let users choose what belongs in the report before asking how individual elements should behave.

3. **Handle relationships automatically**
Don't require users to understand joins or underlying data relationships when the application already knows them.

4. **Show the result as it develops**
Keep report construction and report output closely connected.

5. **Reveal complexity progressively**
Keep advanced filtering, calculations, formatting and configuration available without making them prerequisites for basic reporting.

6. **Default intelligently, override deliberately**
Make safe decisions automatically and give users control when they need something different.

Design Opportunity

The competitive analysis reframed the problem.

The opportunity was **not** to create a simpler version of a traditional report writer. Doing that would still require nontechnical users to learn the mental model of professional reporting software.

The opportunity was to change the mental model itself:

> ***Users should be responsible for describing the information they need. The Report Writer should be responsible for translating those choices into a valid report.***

That became the foundation for the redesign.