Spark Dataset and Model Readiness

> **Alchemer Dashboard is currently waitlist only.** Visit this page **to learn more about Dashboard or join the waitlist!**

Alchemer Dashboard's AI tools achieve accuracy through four primary channels: our AI model training, system post-processing, customer data preparation, and coaching of the AI tool. By taking the time to prepare your data for use with our AI systems, you not only achieve greater accuracy, but increase the intuitiveness and satisfaction your users have with its experience.
There are several proactive steps you can take to optimize Datasets or Models and significantly improve Spark accuracy. These steps include:

Reducing Worksheet or Model size to fewer than fifty columns. Using human-readable and unique column names. Turning on **Indexing** for relevant columns. Adding synonyms for columns that have other nicknames or internal terms.

This article details each step of setting up a Worksheet or Model, and will help your company reach the highest accuracy with our AI systems.

Guidelines for column names

Avoid similar column names
If identical strings appear in multiple columns, it may confuse Spark's judgment on column selection.

Example
For the query 'Show customers by Product', it is difficult for Spark to understand which column to choose between 'Product Line' and' Product Type', and the answer generated may not use the column expected.

Use user-friendly names over abbreviations & domain specific terms
When creating column names, avoid using abbreviations and internal jargon or acronyms, as they may not be easily understood by Spark.

Example
For instance, for the query 'Show the number of policies created last month' , it is difficult for Spark to know whether to choose the 'plc_date' or 'crt_date' column to generate the answer.

Avoid mismatch between column names and datatype
If column names do not align with their respective data types, it can impact accuracy.

Example

If a column named ‘weeks’ is marked as an attribute column with values such as ‘Week 1’ and ‘Week 2’, it makes it difficult for Spark to respond correctly to date related queries. If you want to keep the data, then we suggest renaming the column to ‘Week number’ to make it less confusing for Spark.

Avoid mismatch between column values and data type

Ensure column values match their respective data types.

Example

If a column named 'order_date' has values like '01 Jan 20' but is set to text data type, change it to date data type to improve accuracy.

Use spaces as delimiters in column names

Avoid using other types of delimiters such as brackets and emojis as they may interfere with Spark functionalities.

Avoid overlap in column names and keywords

Avoid overlap between column names and Alchemer Dashboard Full Keyword Reference as it can impact accurate query interpretation.

Example

If a column is named 'Sales growth,' indicating annual sales growth, and a user searches for 'What is the Sales growth month on month' Spark might prioritize the 'Sales growth' column instead of recognizing the user’s intent to use the 'growth' keyword on the ‘Sales’ column in a monthly context.

Avoid column names that start with numbers

You can still name columns with numbers in the middle of the title.

Avoid using special characters (like ‘]’, ‘&’ so on) in the column/formula names

Some special characters might be necessary for better readability such as Profit % to denote percentage of profit calculation. Unless required for performance, we suggest to avoid having such columns and rely on Spark to use the appropriate columns to generate such calculations.

Guidelines for column descriptions

Adding clear and concise column descriptions improves Spark’s accuracy by helping it interpret the column’s meaning, values, and intended use in queries.

Provide meaningful descriptions to improve query interpretation

Include a brief but informative description to clarify what the column represents. This helps Spark generate more accurate responses.

Example

A column labeled "ACV" can have the description: "Annual Contract Value, representing total revenue from contracts over a year." This ensures Spark correctly interprets queries related to revenue and ACV.

Pass contextual information to refine filtering logic

Help Spark apply the correct logic when filtering based on column values.

Example

A column containing date-month values such as 'Jan', 'Feb', etc., can have the description: "Represents date months in 'mmm' format (e.g., Jan). Use when filtering by month." This ensures Spark correctly maps ‘January’ to ‘Jan’ in queries.

Use descriptions to guide Spark on how to use or avoid a column

If a column should be used in a specific way (or avoided in certain contexts), mention this explicitly in the description.

Example

A column with Boolean values can have the description: "Indicates whether an account is active (TRUE = active, FALSE = inactive). Use for filtering active accounts." This prevents misinterpretation of TRUE/FALSE values in queries.

Keep descriptions concise (maximum 200 characters)

Spark will only consider the first 200 characters of the description. Ensure critical details are included within this limit for accurate interpretation.

Guidelines for synonyms

Define synonyms for common terms

We recommend you consistently use easily understandable column names that are widely recognized within the business unit or organization. However, there may be occasions where different terms are employed to denote the same data column.

Example

'Sales' typically denotes the column capturing total revenue from item sales, but 'turnover' and 'revenue' may also be used interchangeably for this purpose within your business context. In such instances, we advise you name the column as 'Sales,' which is the most prevalent term, and define ‘turnover’ and ‘revenue’ as synonyms for the ‘Sales’ column.
This approach will assist Spark in picking the column correctly when users ask for sales, turnover or revenue.

Avoid overlap in synonyms and column names

Ensure that the synonyms for a particular column are clear and distinct from any other column name or column synonyms to avoid confusing Spark from picking the right column.

Example

For the query ‘Show the total expenses for last month’, if there is a column named 'Costs' with a synonym 'Expense', and another column named 'Material Expenses' in the Worksheet or Model, Spark might not be able to select the right column to generate a response

Guidelines for date columns

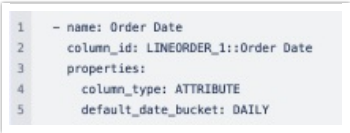
Avoid adding multiple date columns

We advise keeping the number of date columns to a minimum, adding only those that are necessary. Keywords such as 'growth' often rely on date columns, making it challenging for the system to select the appropriate column to generate a response.

If your worksheet has more than 2 date columns, we suggest building content such as Charts and Dashboards containing the specific date columns. This helps Spark understand the relationship between the columns in the Worksheet or Model and can significantly improve Spark’s ability to accurately select the correct date columns when generating a response.

Default aggregation granularity in TML file

If you want to assign a different default aggregation granularity, you can set it for each date column in the Worksheet/Model TML file using the default_date_bucket property.



```
- name: Order Date
  column_id: LINEORDER_1::Order Date
  properties:
    column_type: ATTRIBUTE
    default_date_bucket: DAILY
```

Example

If you set the default_date_bucket to DAILY for the column Order Date, the column is now interpreted as Order Date daily. Any questions such as "Display the trend of orders" would show the trend on a daily level by default.

Guidelines for indexing columns

Ensure the columns being queried upon Spark are indexed

Indexing is essential for enabling access to relevant data values and accurately identifying specific column values in queries.

Example

For instance, consider a query such as ‘Show sales for Metformin’. Without indexing, it becomes difficult for Spark to discern whether to look for the value ‘Metformin’ in the column named ‘Area’ or ‘Vendor Name’ or ‘Drug Name’.

If you don’t see value suggestions in Search Data, then Spark will likely be unable to pick those values correctly.

> Always disable indexing for Personally Identifiable Information (PII) Data

Avoid indexing unused columns with overlapping values
Index only required columns. Too many values from columns might confuse Spark in selecting the right column.

Example
Consider a query such as 'Show sales for Washington area'. Spark may find it challenging to determine whether the user is referring to 'Washington' within the context of the column names 'District' or 'State', as the value is present for both columns. If it's unavoidable in your business context to disable indexing on a column as users can query for values on it, we suggest specifying the column name while querying to help Spark in picking the right column.
For instance, changing the query to 'Show sales for Washington State', will provide a more accurate result.

Avoid indexing descriptive text columns
This can impact accuracy as it confuses Spark in picking the right column.

Ensure indexing of low cardinality columns
Proactively identify and address low cardinality columns lacking indexing to optimize data accessibility and analysis accuracy.

Create formulas or sets for high cardinality columns
This helps in mitigating potential indexing issues and enhancing Spark's accuracy.

Using Optimize for Spark to index attribute columns

To improve search accuracy it's important to index Worksheet and Model attribute columns properly, so Alchemer Dashboard can effectively retrieve and provide sample values associated with those columns. These sample values, combined with the column name, are passed to the LLM, which enables it to do the following:

- Understand the column properties better
- Map the requested value in the query to the correct column value when generating an answer

Optimize for Spark makes this a lot easier because it tells you columns which are not configured to index values and allows you to easily configure indexing on the appropriate columns.

> You must have permission to edit tables in the Model or Worksheet to use *Optimize for Spark*.

To use Optimize for Spark, do the following:

1. In the Data workspace, click the name of the Worksheet or Model you want to optimize for Spark.

2. In the Model or Worksheet, click the **More** menu, and select **Enable Spark**.

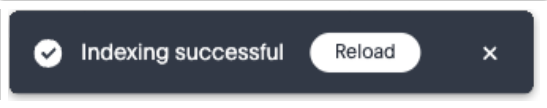
3. Click the **Optimize for Spark** button.

The *Optimize for Spark* window appears listing the columns suggested for indexing.

4. Do one of the following:

- If you want to use the columns that are selected, click **Optimize**.
- If you want select different columns, select only the checkboxes for the columns you want, and click **Optimize**.

  After all selected columns are indexed, the *Indexing successful* message appears at the bottom of the page.



Related Articles