Lab 3. Tableau Extracts

In this lab, we will cover the followingtopics:

- Overview of different file formats in Tableau
- Creating a data source extract
- Configuring an incremental extract
- Upgrading to Hyper
- Creating extracts using cross-database joins
- Troubleshooting extracts with Tableau Server

Introduction

This lab will cover how the Tableau dashboard performance is boosted by using extracts. We'll look at different types of Tableau file formats and types of extracts. The lab introduces us to Tableau's in-memory fast-data-engine technology, called Hyper, that was released in October 2017, and we'll discuss what updates to expect with version 2018.3. This lab will enable us to optimize performance of their Tableau dashboards using aggregated extracts, dimension reduction, extract filters, incremental extract refreshes, and cross-data joins. The principles we use to connect to text files, although very simple examples, can apply to more sophisticated data sources.

Overview of different file formats in Tableau

We'll look at the different file formats used in Tableau: `Tableau Workbooks` (`*.twb`), `Tableau Bookmarks` (`*.tbm`), `Tableau Packaged Workbooks` (`.twbx`), and `Tableau Extracts` (`*.hyper or .tde`).

How to do it..

In this recipe, we'll learn about the different file formats in Tableau. Tableau Extracts will be introduced in this recipe and covered in more detail in the [*Creating a data source extract*] recipe.

Tableau Workbook (TWB)

A [**Tableau Workbook**] ([**TWB**]) is created by the Tableau Desktop. They hold the analytics you create as a collection of one or more of the following: worksheets, dashboards, or stories. Workbooks help to organize your insights. Users can view workbooks on a desktop or server. If sharing these workbooks, the recipient must have access to the data source being used. Workbooks are saved with the `.twb` extension.

To save a workbook, go to `File | Save As` . Provide a name for the `File name` field and select `Tableau Workbook (*.twb),` as shown in the following screenshot:
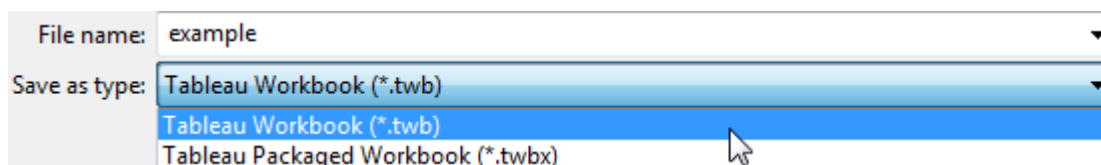
Tableau Packaged Workbook

Workbooks often reference external resources, such as data sources, images, data files, or extract files. When you save a workbook to share with others, it's best to save a packaged workbook instead. Packaged workbooks are made of the workbook and the external resources. They are saved with the `.twbx` extension. Packaged workbooks can be viewed using a desktop, server, or reader.

To save a [**Tableau Packaged Workbook**] ([**TWBX**]), go to `File | Save As`. Provide a name for the the `File name` field and select `Tableau Packaged Workbook`:
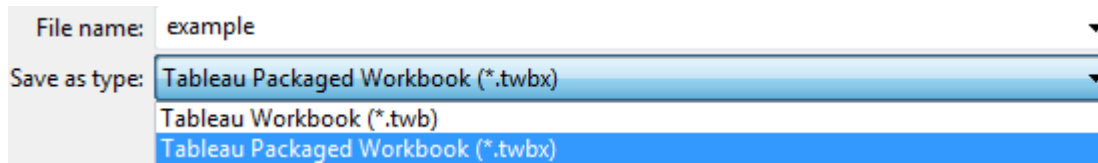

Tableau Bookmark

[**Tableau Bookmark**] ([**TBM**]) can be used to save the individual worksheet of a workbook. It can include data connections and formatting. The major benefit of bookmarks is that they allow two or more developers to work on the same data, where they can then collate their worksheets to a single workbook. Bookmarks can be accessed from any workbook using the `Bookmark` menu if they are saved in the Tableau repository:

1. To save a bookmark, go to `Window | Bookmark | Create Bookmark`:
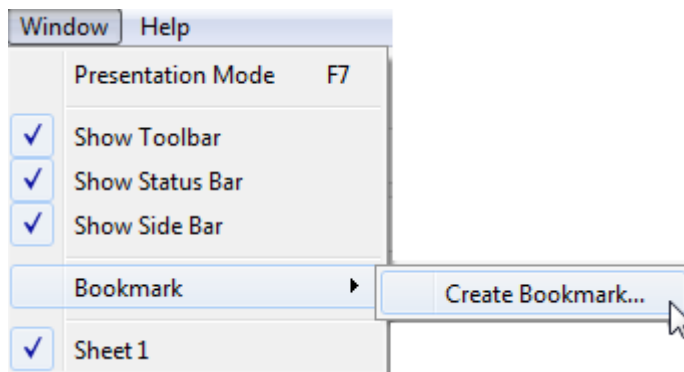
2. To access a bookmark, go to `Window | Bookmark` and find them listed under the `Create Bookmark...` option:
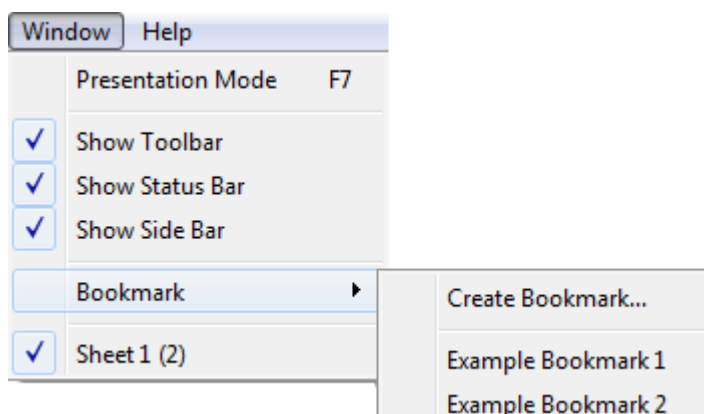
Tableau Data Extract

[**Tableau Data Extract**] ([**TDE**]) contains a compressed local snapshot of the data pulled from any data source based on the join and extract filter conditions specified. Under the hood, it's a columnar store, which makes aggregation and column access very quick. Generally, this will be easier to work with than live-querying the database for every

change. Data extracts can be published to the server so they can be used by others and scheduled to update on a regular cadence. We'll go into detail of how to create an extract in the [*Creating a data source extract*] recipe.

Hyper

[**Hyper**] is a new the technology used in Tableau's data engine since 10.5. It allows users to build bigger extracts more quickly than traditional data extracts. Users dealing with large volumes of data can benefit from upgrading. In 2018.3, we see a new feature that allows users to choose how their data is stored, via a single-table or multiple-table extract option. Multiple-table extracts can potentially improve performance and reduce the extract size. There are conditions on when multiple-table extracts can be used, detailed in the [*Creating a data source extract*] recipe.

How it works...

In this section, we will look briefly at the TWB, TWBX, and TBM file formats.

Tableau Workbook

We saved a workbook by using `File` and `Save As` . We provided a filename for the `File name` field and selected `Tableau Workbook` .

Tableau Packaged Workbook (TWBX)

We saved a [**Tableau Packaged Workbook**] ([**TWBX**]) by using ** `File` **and `Save As` . We provided a a filename for the `File name` field and selected `Tableau Packaged Workbook` .

Tableau Bookmark (TBM)

We saved a [**Tableau Bookmark**] ([**TBM**]) bookmark by going to the `Window` menu. We chose `Bookmark` and then selected `Create Bookmark...` .

We accessed the bookmark by going to the `Window` menu. We chose `Bookmark` and `Create Bookmark...` , and then we selected the bookmarks listed within the menu.

Creating a data source extract

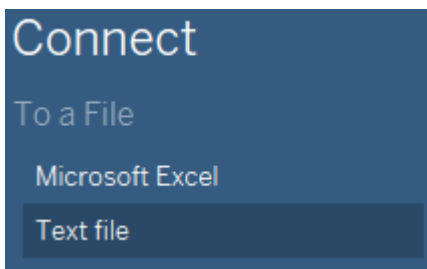In this recipe, we'll learn how to create a data source extract.

How to do it..

We'll quickly go through an example of connecting to data in text files in order to create the extract.
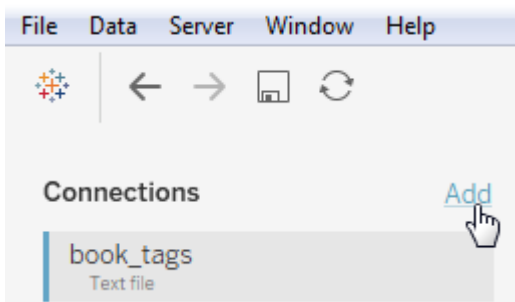
Connecting to data

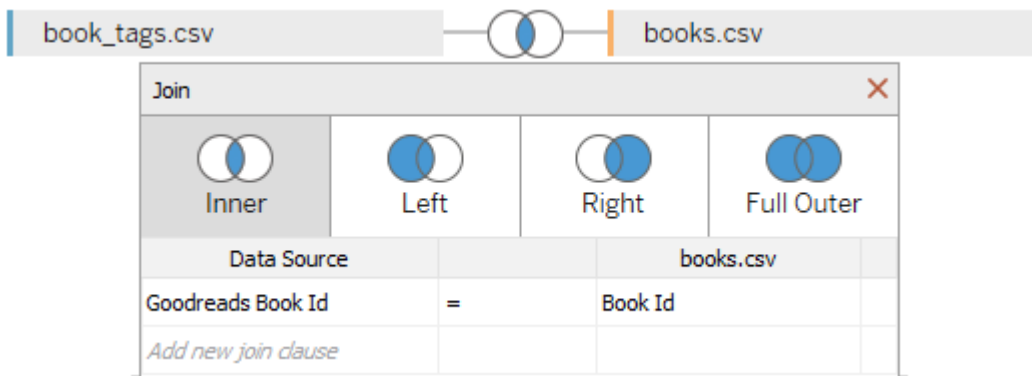To follow along in this example, connect to `books_tags.csv` and `books.csv` , and then perform the following steps:

1. Use `Text file` to connect to `books_tags.csv` :

2. Add another text file connection to `books.csv`:
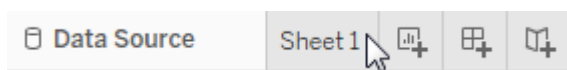

3. Join on `Goodreads Book Id` and `Book Id`:

Creating an extract

After we've successfully connected to the data, we can create an extract, which can be accomplished in multiple places within Tableau. Here, we'll describe a method from the `Data Source` tab:

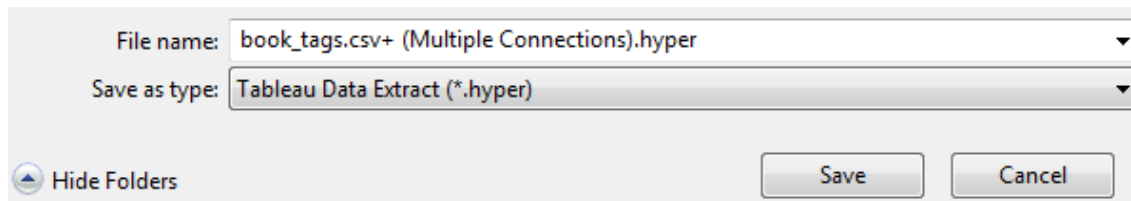1. On the `Data Source` tab, select the `Extract` radial button:

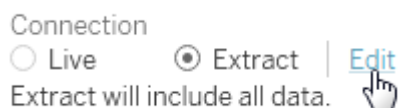2. Click on the `Sheet 1` tab to initiate the extract:

3. Choose a name for the file and click on `Save` . Here, we have chosen the defaults:

Optional settings for extracts

1. There are several options to optimize how the data should be stored, which can help with extract sizes and refresh times. Use the `Edit` link for the menu:

2. We canconfigure how the data should be stored by considering the following two available options:

   ○ `Single table` : This option is the default structure used by Tableau. Select the** `Single table` **option if the amount of data should be limited with extract filters, aggregations, limiting rows, or when using pass-through functions.

   ○ `Multiple tables` : Select `Multiple tables` if your extract has one or more equality joins and the data types of these columns are identical. The options to specify how much data to extract are not valid with `Multiple tables` and will be grayed out:
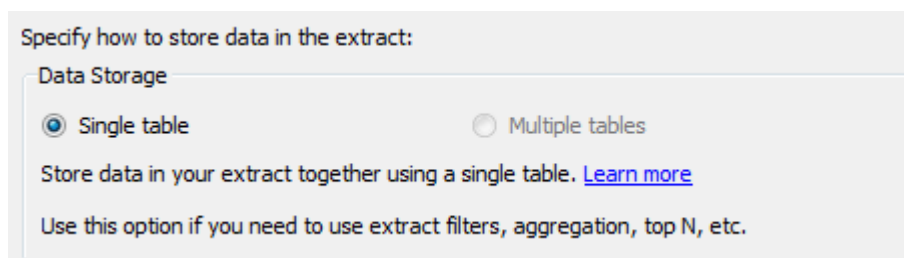
Note

When choosing the `Single table` storage, go to [*step 2*]. When choosing `Multiple tables` storage, go to [*step 5*].

3. For `Single table` storage, it is optional to add filters, select `Add...` to limit the data based on a column and value:
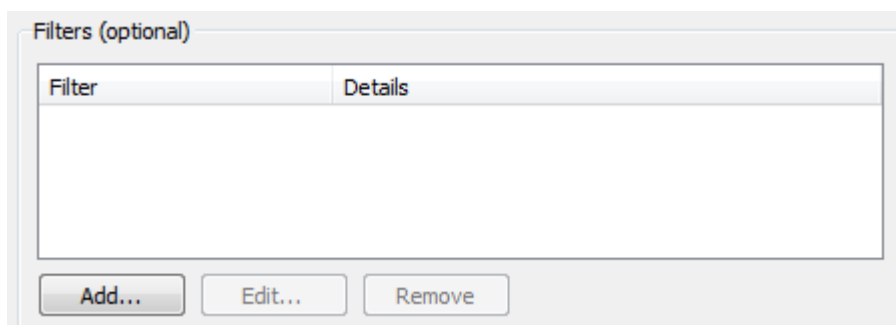
4. Select **Aggregate data for visible dimensions** to aggregate the data, which consolidates the number of rows that can have a positive impact on performance:
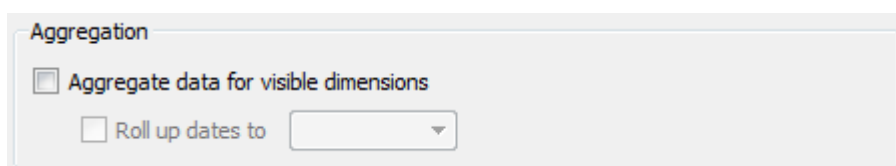
5. Select the **Number of Rows** option specify how many rows are returned in the extract. Tableau will first apply filters and aggregations, and then extract the number of rows specified. Some data sources don't support sampling, so this option may not be available:
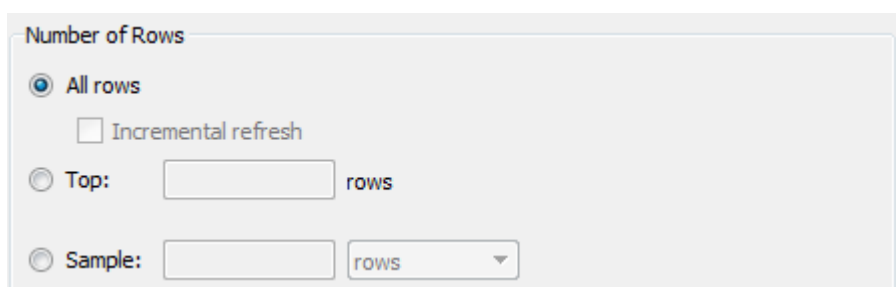
6. The **Hide All Unused Fields** option is optional. Select this option to exclude the hidden fields on the sheets or the **Data Source** tab from the extract:

7. Save the extract by following the directions within the [*Creating an extract*] section.

Note

The workbook will automatically use the extract once it's created. The connection to the data extract will not persist when it's opened again unless the workbook is saved.

How it works...

We created a local extract by navigating to the **Data Source** tab and selecting **Extract** . To initiate the **Extract** , we click on **Sheet 1** and **Save** .

`Edit` provides options for data storage since version 2018.3, and limits data by filtering, aggregating, limiting rows, or hiding fields. These options help optimize the extracts by limiting their size and refresh times.

There's more...

Locally, once the data source is extracted, we can update it by selecting `Refresh`, as shown in the following screenshot:

With Tableau Server, as long as the server has rights and access to the data source, we can publish the extract and assign it to a refresh schedule. Updating the workbook to reference the published extract will refresh the data per a defined schedule instead of manually.

Note

In situations where data needs to be accessed offline, using a local extract is better than using a published version.

Extract encryption at rest is a new data feature that allows users to encrypt their extracts on Tableau Server. Admins can enforce this policy on all extracts or allow users to choose. Other Tableau tools like Desktop or Prep will not have encryption features.

Configuring an incremental extract

By default, extracts will update using a full refresh, meaning all of the rows are replaced during a build. Depending on the size of the extract and whether there is a mechanism to identify new rows, incremental extracts can help provide users with new data more quickly. Rather than rebuilding the entire extract, for example, each day we can add the new rows based on date instead.
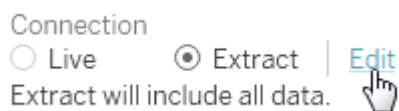
Getting ready

Use Tableau 2020.x, make sure you have a data source ready.

How to do it..

Rather than refreshing the entire extract each time, we can configure a refresh to add only the rows that are new since the previous refresh:

1. Create a data source using a `Text file` connection to `madrid_2017.csv`.
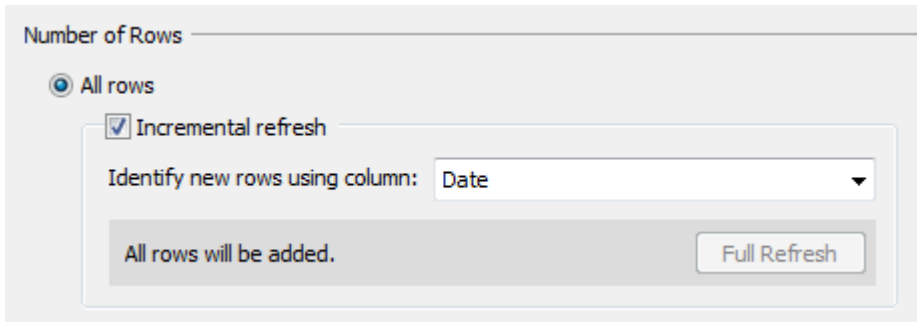2. Select the `Extract` radio button and click on `Edit`:

3. Select `All rows` under `Number of Rows` to extract:
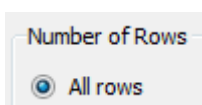
Note

`Incremental refresh` can only be defined when you are extracting as** `Single table` ** with the `All rows` in database.

4. Select `Incremental refresh` and then specify a column that identifies new rows:
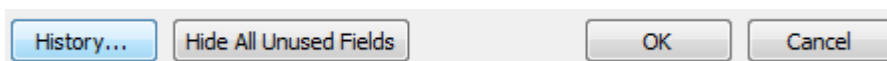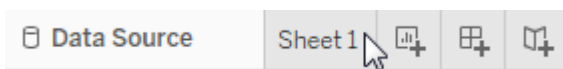
5. Click on `OK` and save the `.hyper` file:

6. Click on `Sheet 1` and give the extract a name:

Note

These steps can be used for new or existing extracts. When editing an existing extract, the last refresh is shown.

7. To verify the latest date loaded, add `Date` to the `Columns` shelf.
8. Show the maximum value by right-clicking on `Date` , choosing `Measure` , and then selecting `Maximum` :

Columns   ▼   MAX(Date)   ▼

Rows

Filter...

Show Filter
Show Highlighter

Sort...
Format...
✓ Show Header
✓ Include in Tooltip

Year	2015
Quarter	Q2
Month	May
Day	8
More	▶

Year	2015
Quarter	Q2 2015
Month	May 2015
Week Number	Week 5, 2015
Day	May 8, 2015
More	▶

Exact Date

Attribute

● Measure (Maximum)   ▶

● Discrete
Continuous

Edit in Shelf

✓ Ignore in Table Calculations

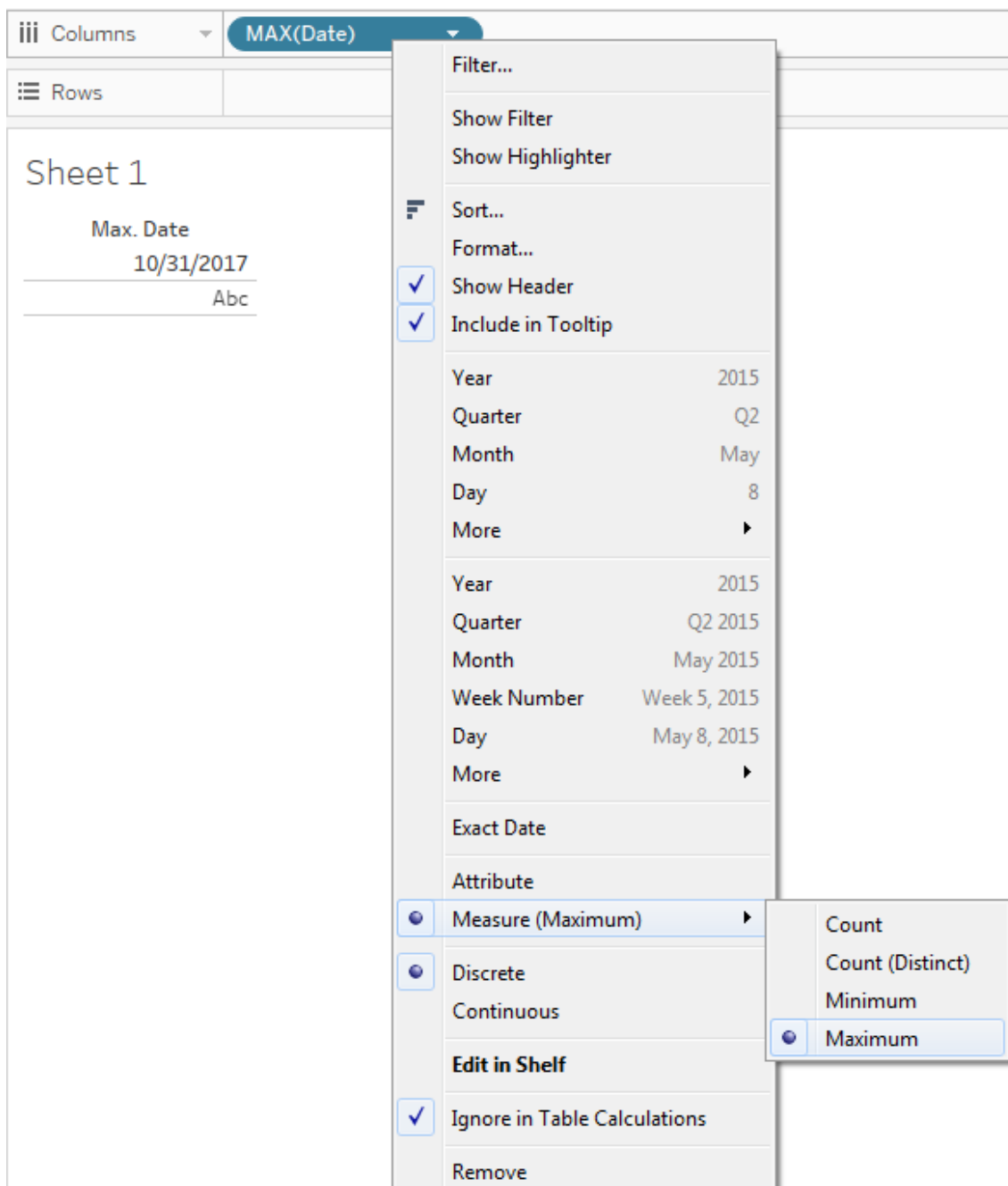Remove

Count
Count (Distinct)
Minimum
● Maximum

Sheet 1

Max. Date

10/31/2017

Abc

9. Save the workbook and close Tableau.

10. To simulate the incremental load, copy all the data from `Nov.csv` and paste it at the end of `madrid_2017.csv` . Save the file.

11. Open the workbook and click on `Refresh` :

Connection
◯ Live   ◉ Extract | Edit   Refresh

12. Go back to `Sheet 1` , we now see November 30 as the `Max. Date` :

Max. Date
11/30/2017
Abc

How it works...

We selected the `Incremental refresh` checkbox and identified `Date` as the column Tableau would use to check for the presence of new rows.
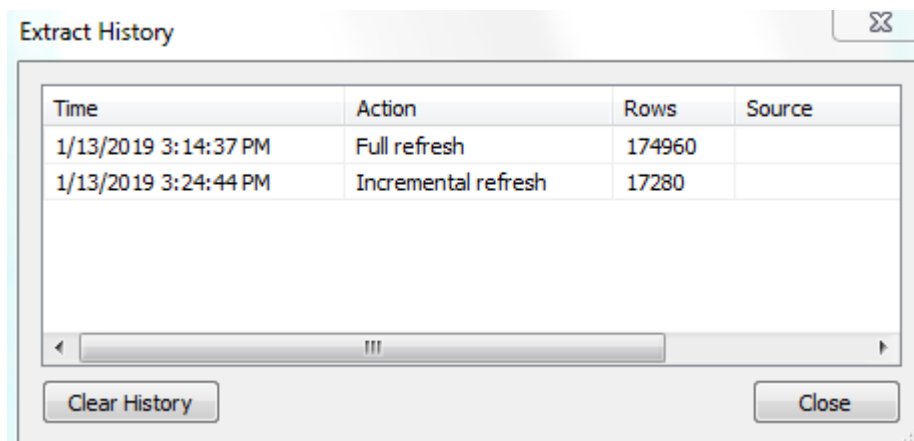
There's more...

View the refresh history for an extract by selecting a data source on the `Data` menu and then select `Extract` and `History...` :

| History... | Hide All Unused Fields | | OK | Cancel |
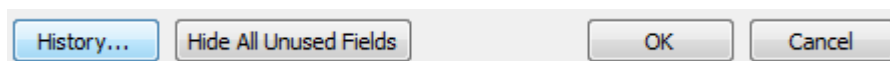
The `Extract History` window shows the date and time of each refresh, whether it was full or incremental, and the number of rows processed. If the refresh was from a file, it also shows the source filename.

We can see all these details in the following screenshot:

Extract History

Time	Action	Rows	Source
1/13/2019 3:14:37 PM	Full refresh	174960	
1/13/2019 3:24:44 PM	Incremental refresh	17280	

Clear History   Close

Upgrading to Hyper

Since 10.5, extracts use the `.hyper` format instead of `.tde` . Hyper is an improved data engine that allows for bigger extracts in the billions of rows, faster extract creation, and better performance. This section describes the steps to upgrade the `.tde` extract to the `.hyper` extract. It also details the compatibility considerations since Tableau Desktop, Online, and Server do not upgrade simultaneously.

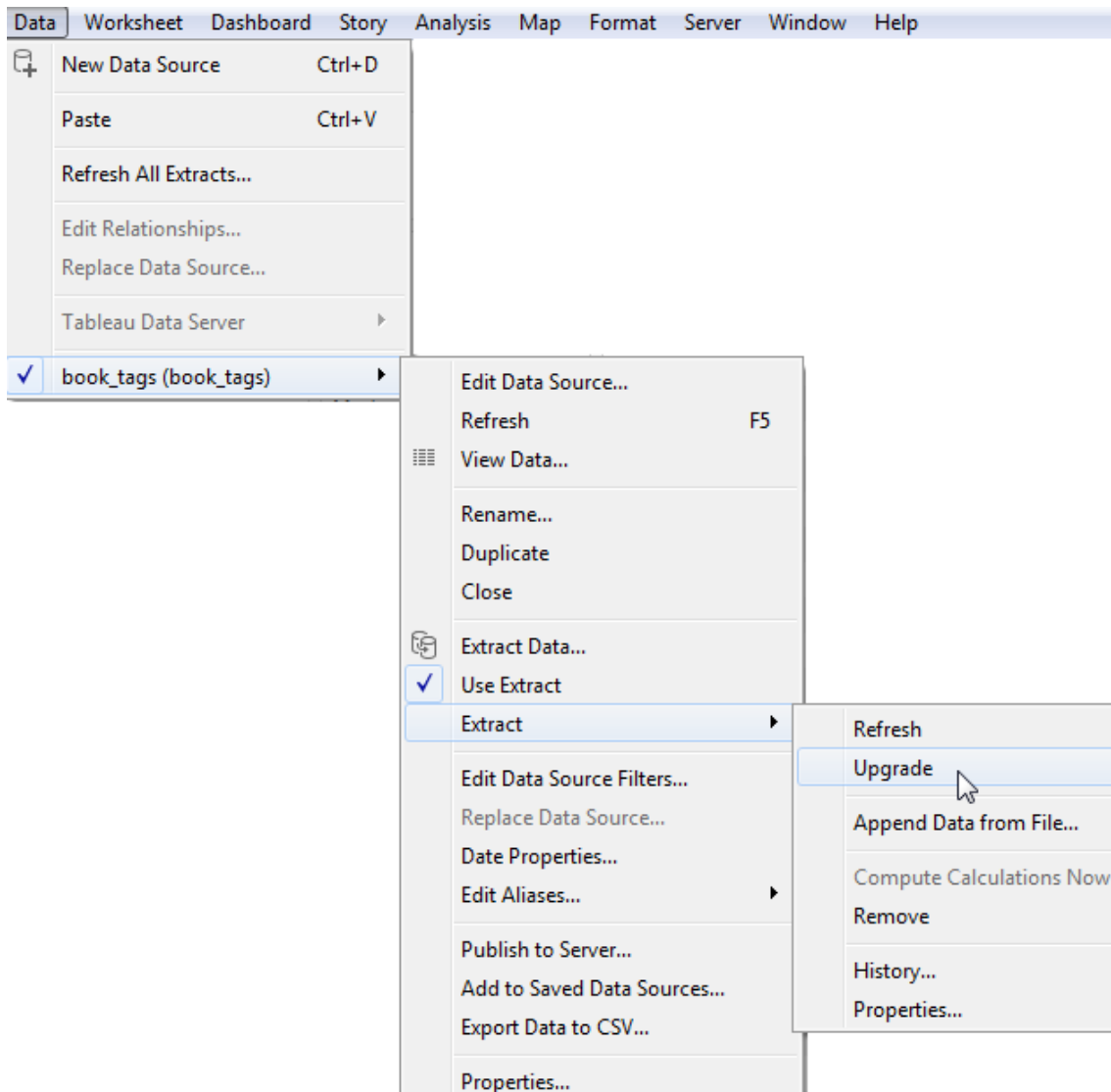## Getting ready

In this recipe, we'll learn how to upgrade the `.tde` extracts to `.hyper` .

How to do it..

If using local extracts, use Tableau Desktop to manually upgrade a `.tde` extract to a `.hyper` extract.

1. Open a workbook that uses a `.tde` extract.
2. From the **Data** menu, select **Extract | Upgrade** :

3. Select **File | Save** to complete the extract upgrade. Outside of Tableau Desktop, extracts can be upgraded in the following way from Tableau Server or Tableau Online:

- Manual refresh
- Scheduled full or incremental extract refresh

How it works...

We opened a workbook using a `.tde` extract in the latest version. We upgraded the extract and saved the file.

Creating extracts using cross-database joins

This section details the steps involved in creating table joins from separate data sources and extracts. Cross-database joins are helpful when you need to analyze data from two or more different databases as if they were one. Typically, a technology team resource would be needed to stage and integrate this data in a single place.
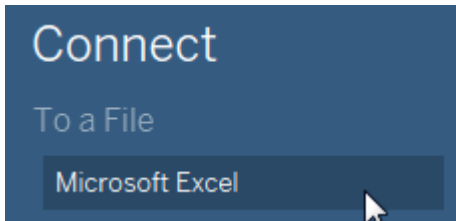
Getting ready

In this recipe, we'll learn how to perform a cross-database join.

How to do it..

Cross-database joins allow users to engage with the data in Tableau as if it lives in a single source.

To follow along, use `book_tags.xlsx` and `books.csv`, and then perform the following steps:

1. Open Tableau and create a **Microsoft Excel** data source connection to `book_tags.xlsx`:

2. Use **Add** to bring in another data source:
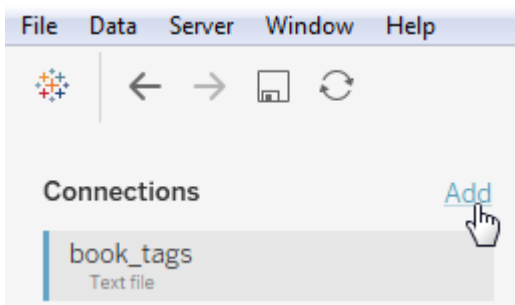
3. Connect to the `books.csv` text file, which is a different data source:

4. Join on the key fields, **Goodreads Book Id** and **Book Id**:

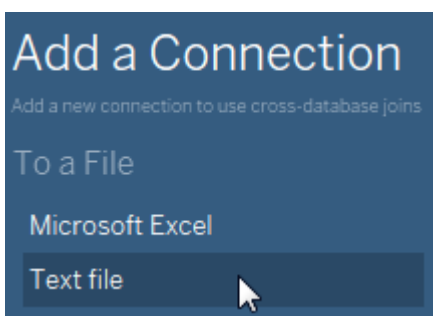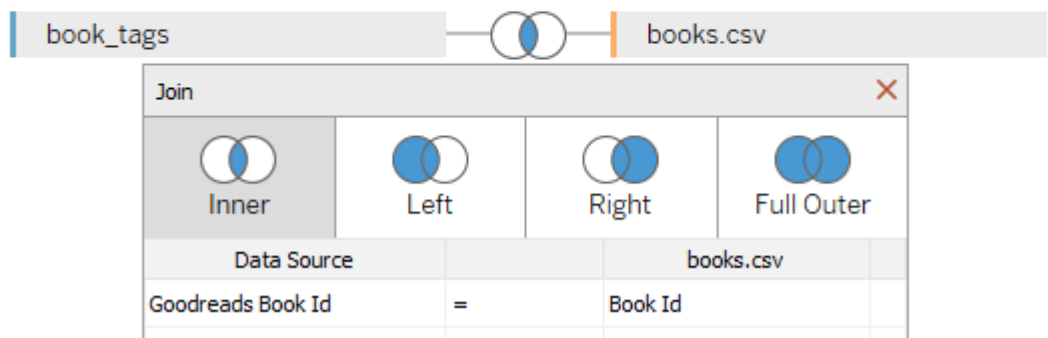# book_tags+ (Multiple Connections)

book_tags — books.csv

Join				X
Inner	Left	Right	Full Outer	

Data Source		books.csv	
Goodreads Book Id	=	Book Id	

Note

The fields used in a cross-database join should be the same data type. We can use any of the join types. Notice each connection is in a different color:

book_tags+ (Multiple Connections)

book_tags — books.csv

| Sort fields | Data source order | ▼ |

#	#	#	#	#	#	#
book!tags	book!tags	book!tags	books.csv	books.csv	books.csv	books.csv
Goodreads Book Id	**Tag Id**	**Count**	**Id**	**Book Id**	**Best Book Id**	**Work Id**

How it works...

We connected to two data sources of different types: Excel and text. The field used in the cross-database join are of the same data type. We were able to identify columns from each data source by the color assigned to each column and table. We saw blue for the Excel sheet and orange for the text file.

Troubleshooting extracts with Tableau Server

After a `.tde` extract has been upgraded, it can't be opened using an earlier version of Tableau Desktop. If you can't upgrade, you'll need to keep your extract in the `.tde` format.

Getting ready

You'll need an older version of Tableau Desktop installed on your machine and/or the original data source connection.

How to do it..

If you've upgraded by mistake or wish to retain a previous version, consider the following two options:

- Recreate an extract using an earlier version of Tableau Desktop by connecting to the original data source using an earlier version of Tableau Desktop
- Roll back an extract to a previous version, if revision history is turned on for Server or Tableau Online, by downloading the earlier version

How it works...

When we've mistakenly upgraded to a later version, we must roll back to a previous version. You do this rollback by either recreating the work in an older version of Tableau or by getting an older version from the server.

There's more...

If you need to keep an extract in the `.tde` format, don't upgrade the extract; keep an earlier version of Tableau Desktop for extracts, refreshes, or appending data. Also disable extract refreshes on the server for any that should remain in the legacy format. See the [*Hyper*] section to understand which actions can cause an extract to upgrade.

See also

Consider upgrading to 2018.x or later so the benefits of Hyper extracts can be utilized across all platforms.

When upgrading, we should understand the following backward-compatibility limitations:

- The `.hyper` files can't be downgraded to the `.tde` extracts
- Older versions of Tableau Desktop can't be published to an upgraded server environment
- Upgraded extracts can't be opened in an earlier version of Tableau, which could limit how they're shared with others who are using older versions
- **Export as a Version** can't be used to downgrade a workbook with the `.hyper` extracts