Bibliography in LaTeX with `biblatex`

This LaTeX document uses the `biblatex` package to manage citations and bibliographies. Below, I'll explain every relevant part of the bibliography setup.

---

1. Including the `biblatex` Package

```
\usepackage[style=ieee, sorting=ynt]{biblatex}
```

- `style=ieee` : This sets the bibliography style to **IEEE** format.
- `sorting=ynt` : Controls how bibliography entries are sorted:
  - `y` → Sort by **year** (oldest first).
  - `n` → Sort by **name** (author/editor).
  - `t` → Sort by **title**.

`biblatex` is more powerful than traditional `bibtex` as it allows more flexible citation formatting.

---

2. Adding the Bibliography File

```
\addbibresource{citations.bib}
```

This tells LaTeX to use `citations.bib` as the **bibliography database**. This file (not shown in your code) should contain entries like:

```
@book{norman2013design,
  author = {Donald A. Norman},
  title = {The Design of Everyday Things},
  year = {2013},
  publisher = {Basic Books}
}

@book{twain2008tom,
  author = {Mark Twain},
  title = {The Adventures of Tom Sawyer},
  year = {2008},
  publisher = {Penguin Classics}
}

@article{wittern2016javascript,
  author = {Ekaterina Wittern and others},
  title = {JavaScript: The Good, the Bad, and the Ugly},
  year = {2016},
  journal = {IEEE Software}
}

@online{wiki2025quaternion,
  author = {Wikipedia},
  title = {Quaternion Mathematics},
  year = {2025},
  url = {https://en.wikipedia.org/wiki/Quaternion}
}
```

---

3. Citing Works in the Document

Different citation commands in `biblatex` allow formatting flexibility.

Standard Citations

```
We can cite to a work with \cite{norman2013design}.
```

- Outputs: **[1]** (IEEE style)
- This is a numeric citation referring to `[1]` in the bibliography.

Multiple Citations

```
We can also cite multiple works \cite{norman2013design, twain2008tom, wiki2025quaternion, wittern2016javascript}.
```

- Outputs: **[1, 2, 3, 4]**
- Groups citations into a single set.

Author-based Citation (`\textcite{}`)

```
\textcite{wittern2016javascript} discussed about a horrible language.
```

- Outputs: **Wittern et al. [4] discussed about a horrible language.**
- Used when integrating the author's name into the sentence.

Parenthetical Citation (`\parencite{}`)

```
We can put a citation within a parenthesis with \parencite{norman2013design}.
```

- Outputs: **([1])**
- Works similarly to `\cite{}` but explicitly places the citation in parentheses.

Footnote Citation (`\footcite{}`)

```
Take a look at the wiki of Quaternions \footcite{wiki2025quaternion}.
```

- Creates a citation in a **footnote**.

---

4. Printing the Bibliography

```
\printbibliography[heading=bibintoc, title={Bibliography}]
```

- `heading=bibintoc` : Adds the bibliography to the **table of contents**.
- `title={Bibliography}` : Sets the section title as "Bibliography".

Alternative Method (BibTeX)

The commented-out lines show an alternative approach:

```
% \bibliographystyle{acm}
% \bibliography{citations}
```

- This uses **BibTeX**, an older system.
- `\bibliographystyle{acm}` → Uses ACM citation style.
- `\bibliography{citations}` → Inserts references from `citations.bib` .
- Requires running `bibtex` separately, whereas `biblatex` works directly with `biber` .

---

5. Differences Between `bibtex` and `biblatex`

Feature	`bibtex`	`biblatex`
Citation Styles	Uses `\bibliographystyle{}`	Uses `style=` option
Bibliography Command	`\bibliography{}`	`\printbibliography`

Feature	Processing Tool Needs `bibtex` `bibtex`	Uses `biber` `biblatex`
Flexibility	Less customizable	More flexible formatting
Multiple Bibliographies	Harder to manage	Fully supported

Recommendation: `biblatex` is better for modern LaTeX documents.

---

6. Summary of Commands

Command	Usage
`\cite{key}`	Numbered citation (e.g., `[1]`)
`\parencite{key}`	Parenthetical citation (e.g., `([1])`)
`\textcite{key}`	Inline author citation (e.g., `Author [1]`)
`\footcite{key}`	Footnote citation
`\addbibresource{file.bib}`	Adds a bibliography file
`\printbibliography`	Prints bibliography

---

7. Final Thoughts

Your LaTeX document is correctly structured for using `biblatex` with IEEE-style citations. If you're using Overleaf or compiling locally:

1. Compile using **PDFLaTeX**.
2. Run **Biber** to process citations.
3. Recompile with **PDFLaTeX**.

Would you like a more customized bibliography layout or citation style? 😊

In **LaTeX**, you can change font size in multiple ways depending on whether you want to adjust it **locally (for specific text)** or **globally (for the whole document)**.

---

1. Changing Font Size Locally

Use LaTeX font size commands inside braces `{}` to apply them to specific text.

```
{\tiny Tiny Text}
{\scriptsize Script Size Text}
{\footnotesize Footnote Size Text}
{\small Small Text}
{\normalsize Normal Size (Default)}
{\large Large Text}
{\Large Larger Text}
{\LARGE Even Larger Text}
{\huge Huge Text}
{\Huge Hugest Text}
```

Example:

```
This is {\Large a large text} inside normal text.
```

---

2. Changing Font Size Globally

Modify the font size for the whole document using **document class options**.

```
\documentclass[12pt]{article}  % Other options: 10pt (default), 11pt
```

Alternatively, for section titles and other elements, use `\renewcommand{\...}`:

```
\renewcommand{\normalsize}{\large}  % Changes normal text to \large
```

---

3. Changing Font Size for a Whole Section

To change font size for a larger block of text:

```
\begingroup
\Large
This entire block is in large font.
\endgroup
```

Or use the `\fontsize{size}{baselineskip}\selectfont` command:

```
\fontsize{14pt}{16pt}\selectfont This text is 14pt.
```

LaTeX Notes: Optional Arguments, Preamble, Commands, and Visual Hierarchy

1. Optional Arguments in LaTeX

What are Optional Arguments?

Optional arguments in LaTeX allow customization of commands by passing additional parameters. They are enclosed in square brackets `[ ]`, while mandatory arguments are enclosed in curly braces `{ }`.

Example:

```
\documentclass[12pt]{article}  % '12pt' is an optional argument for font size
```

Here, `12pt` is an optional argument that modifies the document class.

Another example with `\includegraphics`:

```
\includegraphics[width=0.5\textwidth]{image.png}  % Optional argument for width
```

Here, `width=0.5\textwidth` is an optional argument specifying the image size.

2. The Preamble in LaTeX

What is the Preamble?

The **preamble** is the section at the beginning of a LaTeX document where global settings, packages, and configurations are defined. It starts before `\begin{document}`.

Example Preamble:

```
\documentclass[12pt]{article}
\usepackage{graphicx}  % Package for including images
\usepackage{amsmath}   % Math symbols
\usepackage[colorlinks=true, urlcolor=blue]{hyperref}  % Hyperlinks
```

This preamble sets font size, imports required packages, and enables hyperlinks.

3. Using Arguments in LaTeX Commands

What are Commands Called in LaTeX?

In LaTeX, commands are formally called **control sequences**. They begin with a backslash (\) and can accept arguments.

Example of Commands with Arguments:

```
\textbf{Bold Text}  % Command with a single argument
\section{Introduction}  % Section command with title as argument
```

Commands with Optional and Mandatory Arguments:

```
\documentclass[12pt]{article}  % '12pt' is optional, 'article' is mandatory
\usepackage[margin=1in]{geometry}  % 'margin=1in' is optional
```

Some commands support both optional and mandatory arguments:

```
\newcommand{\example}[2][Default]{#1 and #2}
\example{Required}  % Uses 'Default and Required'
\example[Custom]{Required}  % Uses 'Custom and Required'
```

Here, the first argument `[ ]` is optional with a default value, and `{ }` is mandatory.

4. Margins in LaTeX using `geometry`

What is `geometry`?

The `geometry` package in LaTeX allows you to control page margins precisely. You can set uniform margins or specify individual ones for top, bottom, left, and right.

Basic Usage:

```
\usepackage[margin=1in]{geometry}  % Sets 1-inch margins on all sides
```

Custom Margins:

You can specify each margin separately:

```
\usepackage[top=1in, bottom=1.5in, left=1.2in, right=1in]{geometry}
```

This sets different values for each margin.

Page Size and Advanced Layouts:

```
\usepackage[a4paper, margin=2cm]{geometry}  % A4 paper with 2cm margins
\usepackage[letterpaper, left=1in, right=1.5in, top=1in, bottom=2in]{geometry}
```

This allows fine control over document layout for different paper sizes.

5. Using Templates and Special Document Classes

Using Templates in LaTeX

Templates help streamline document formatting by providing pre-defined structures and styles. Many LaTeX templates are available for reports, articles, CVs, and more.

Example of Using a Template:

1. Download a LaTeX template (e.g., from Overleaf, ShareLaTeX, or a university website).
2. Open the `.tex` file and edit the content within `\begin{document} ... \end{document}`.
3. Compile the document using `pdflatex` or an online LaTeX editor.

A common template structure:

```
\documentclass{article}
\usepackage{graphicx}
\title{My Document}
\author{John Doe}
\date{\today}
\begin{document}
\maketitle
\section{Introduction}
This is an example template.
\end{document}
```

Using IEEE and Other Specialized Document Classes

LaTeX supports specialized document classes for different academic and professional formats.

IEEE Format: IEEE publishes an official LaTeX template for research papers.

```
\documentclass[conference]{IEEEtran}  % IEEE Conference paper format
```

For journal articles:

```
\documentclass[journal]{IEEEtran}  % IEEE Journal paper format
```

More details and templates can be found on the IEEE website.

Other Document Classes:

- `report` – For longer documents with chapters.
- `book` – For books, including chapters, sections, and parts.
- `letter` – For formal letters.
- `beamer` – For presentations.

Example of a Report Document Class:

```
\documentclass[12pt]{report}
\begin{document}
\chapter{Introduction}
This is a sample report document.
\end{document}
```

6. Visual Hierarchy in LaTeX

What is Visual Hierarchy?

Visual hierarchy in LaTeX refers to the structured arrangement of text elements to indicate importance. This includes: - Sectioning commands - Font styles and sizes - Lists and indentation

Sectioning Commands:

LaTeX has built-in sectioning commands to create hierarchy:

```
\section{Main Section}
\subsection{Subsection}
\subsubsection{Sub-subsection}
```

Each level decreases in prominence, helping organize the document.

Font Styling for Emphasis:

```
\textbf{Bold Text}    % Bold
\textit{Italic Text}  % Italic
\underline{Underline} % Underline
```

These help highlight important text.

Lists for Structure:

```
\begin{itemize}  % Unordered list
    \item First item
    \item Second item
\end{itemize}
```

```
\begin{enumerate}  % Ordered list
    \item First step
    \item Second step
\end{enumerate}
```

Lists improve readability and structure.

Conclusion

Understanding optional arguments, the preamble, arguments in commands, `geometry` for margins, templates, IEEE format, and visual hierarchy is essential for mastering LaTeX. Proper use of these features ensures well-structured and visually appealing documents.

The `lstlisting` environment you are using comes from the **`listings`** package in LaTeX, which is designed for including source code with syntax highlighting and customization options.

---

1. Required Package

To use `lstlisting`, you must include the `listings` package in your LaTeX document:

```
\usepackage{listings}
\usepackage{xcolor} % For custom colors (optional but useful)
```

---

2. Understanding the Given Block

Your block:

```
\begin{lstlisting}[language=C++,rulecolor=\color{blue}]
```

Breakdown:

1. **`language=C++`** → Specifies C++ syntax highlighting.
2. **`rulecolor=\color{blue}`** → Changes the frame (rule) color to blue.

---

3. Customizing **`listings`**

You can fully customize `lstlisting` with various options:

Basic Example:

```
\documentclass{article}
\usepackage{listings}
\usepackage{xcolor}

\begin{document}

\begin{lstlisting}[language=C++, rulecolor=\color{blue}, frame=single, basicstyle=\ttfamily]
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
\end{lstlisting}

\end{document}
```

Explanation of Options:

- `language=C++` → Enables syntax highlighting for C++.
- `rulecolor=\color{blue}` → Sets the frame color to blue.
- `frame=single` → Draws a frame around the listing.
- `basicstyle=\ttfamily` → Uses a monospaced font.

---

4. More Customization

You can define a default style using:

```
\lstset{
    language=C++,
    frame=single,
    rulecolor=\color{blue},
    basicstyle=\ttfamily\small,
    keywordstyle=\color{red},
    commentstyle=\color{green},
    stringstyle=\color{orange},
    numbers=left,
    numberstyle=\tiny\color{gray},
    stepnumber=1
}
```

Then, whenever you use `lstlisting`, it will apply these styles automatically.

Let me know if you need more details! The `\lstset{}` command in the `listings` package allows extensive customization of how code listings appear in your LaTeX document. Below is a comprehensive list of all the important options you can use with `\lstset{}`.

General Options

Option	Description	Example
`language`	Sets the programming language for syntax highlighting	`language=C++`
`frame`	Adds a border around the listing	`frame=single`, `frame=shadowbox`
`rulecolor`	Sets the color of the frame	`rulecolor=\color{blue}`
`basicstyle`	Changes the font style of the code	`basicstyle=\ttfamily\small`
`showstringspaces`	Shows spaces inside strings with underscores	`showstringspaces=false`
`tabsize`	Defines the number of spaces per tab	`tabsize=4`
`columns`	Adjusts column formatting	`columns=flexible`

Color and Styling

Option	Description	Example
`keywordstyle`	Style for keywords	`keywordstyle=\color{red}\bfseries`
`commentstyle`	Style for comments	`commentstyle=\color{green}\itshape`
`stringstyle`	Style for strings	`stringstyle=\color{orange}`
`numberstyle`	Style for line numbers	`numberstyle=\tiny\color{gray}`
`backgroundcolor`	Background color of the code block	`backgroundcolor=\color{lightgray}`
`emph`	Emphasizes specific words	`emph={int, float}, emphstyle=\color{blue}`

Line Numbering

Option	Description	Example
`numbers`	Enables line numbers	`numbers=left`
`stepnumber`	Sets line number interval	`stepnumber=1`
`numbersep`	Distance between code and line numbers	`numbersep=5pt`
`firstnumber`	Starts numbering from a specific line	`firstnumber=100`
`resetmargins`	Adjusts margins automatically	`resetmargins=true`

Formatting & Spacing

Option	Description	Example
`breaklines`	Wraps long lines automatically	`breaklines=true`
`breakatwhitespace`	Only breaks lines at spaces	`breakatwhitespace=true`
`escapeinside`	Allows inline LaTeX within code	`escapeinside={(*}{*)}`

Special Features

Option	Description	Example
`caption`	Adds a caption to the code block	`caption=Example Code`
`label`	Adds a reference label	`label={lst:example}`
`morekeywords`	Defines additional keywords	`morekeywords={MyKeyword,AnotherKeyword}`
`firstline`	Starts display from a specific line	`firstline=5`
`lastline`	Ends display at a specific line	`lastline=20`

Example Usage

```
\documentclass{article}
\usepackage{listings}
\usepackage{xcolor}

\lstset{
    language=C++,
    frame=single,
    rulecolor=\color{blue},
    basicstyle=\ttfamily\small,
    keywordstyle=\color{red}\bfseries,
    commentstyle=\color{green}\itshape,
    stringstyle=\color{orange},
    numbers=left,
    numberstyle=\tiny\color{gray},
    stepnumber=1,
    backgroundcolor=\color{lightgray},
    breaklines=true,
    caption=Example C++ Code,
    label={lst:example}
}

\begin{document}

Here is an example of a C++ program:

\begin{lstlisting}
#include <iostream> // This is a comment

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
\end{lstlisting}

\end{document}
```

This will produce a nicely formatted C++ code block with:

- **Syntax highlighting**
- **Custom colors**
- **Line numbers**
- **Automatic line wrapping**
- **A caption**

Let me know if you need more customization!

LaTeX Guide: Mathematical Notation and Formatting

1. Document Structure

Basic Setup

```
\documentclass[12pt, a4paper]{article}  % Defines article class with 12pt font size and A4 p
\usepackage[margin=1in]{geometry}       % Sets 1-inch margins.
\usepackage{stix2}                      % Uses STIX Two font for better typography.
\usepackage[colorlinks]{hyperref}        % Enables clickable links with color.
\usepackage{amsmath, amssymb}            % Provides enhanced math symbols and environments.
```

2. Text Formatting

Quotation Marks

Use double backticks (``) and double apostrophes ('') to generate proper quotation marks:

```
``The teacher said, ``You must take permission.''
```

3. Mathematical Expressions

Math Modes

- **Inline Math:** `$(a + b)^2 = a^2 + 2ab + b^2$`
- **Alternate Inline Math:** `\( (a + b)^2 = a^2 + 2ab + b^2 \)`
- **Display Math (centered):**
  ```
  \[ (a + b)^2 = a^2 + 2ab + b^2 \]
  ```
- **Using `equation` environment:**
  ```
  \begin{equation}
      (a + b)^2 = a^2 + 2ab + b^2
  \end{equation}
  ```

Super and Subscripts

Use `^{}` for superscripts and `_ {}` for subscripts:

```
\begin{equation}
    a^{3.14} + b_{(i, j)}
\end{equation}
```

Fractions & Square Roots

- **Basic fraction:** `\frac{a}{b}`
- **Square root:** `\sqrt{x}`
- **Complex expression:**

```
\begin{equation}
    \sqrt{\frac{(a + b)}{(a - b)}}
\end{equation}
```

Operators & Trigonometry

Use `\sin`, `\cos`, `\tan`:

```
\begin{equation}
    \sin^2\theta + \cos^2\theta = 1
\end{equation}
```

Brackets & Parentheses

Use `\left` and `\right` for automatic resizing:

```
\begin{equation}
    \left( \frac{a}{b} \right)
\end{equation}
```

4. Greek Alphabets & Calculus

Greek Letters

```
\alpha \beta \gamma \delta \Gamma \Delta
```

Limits & Integrals

```
\begin{equation}
    \lim_{x \to \infty} \frac{1}{x} = 0
\end{equation}
```

```
\begin{equation}
    \int_{0}^{100} x dx = 500
\end{equation}
```

5. Calligraphy & Sets

```
\begin{equation}
    \mathcal{ABCDE} \mathbb{ABCDE}
\end{equation}
```

```
\begin{equation}
    A \cup B = C \cap D, \text{ for all } x \in \mathbb{Z}
\end{equation}
```

6. Vectors & Comparison Operators

```
\begin{equation}
    \vec{A} \times \vec{B} = \vec{C}, \quad \hat{i} \times \hat{j} = \hat{k}
\end{equation}
```

```
\begin{equation}
    1 = 1 < 2 > 5 \neq 10 \geq 2 \leq 5
\end{equation}
```

7. Advanced Math Environments

Multiline Equations

```
\begin{multline}
    a + b + c + d + e + f + g + h + i + j \\
    + k + l + m + n + o + p + q + r = 10
\end{multline}
```

Splitting Equations

```
\begin{equation}
    \begin{split}
        x &= \frac{10}{20} \\
        &= \frac{1}{2} \\
        &= 0.5
    \end{split}
\end{equation}
```

Alignment of Equations

```
\begin{align}
    a + 2b - 3c &= 10 \\
    2a + 5b + 6c &= -2 \\
    -a + b + 6c &= -4
\end{align}
```

Matrix Representation

```
\begin{equation*}
    \mathbf{A} = \begin{bmatrix}
        1 & 2 & 3 \\
        4 & 5 & 6 \\
        7 & 8 & 9 \\
    \end{bmatrix}
\end{equation*}
```

Conclusion

This document provides an **easy-to-remember** guide for writing LaTeX mathematical expressions with examples and best practices.

LaTeX Tables Documentation

Introduction

Tables are essential for structuring data in LaTeX documents. LaTeX provides various packages and features to format tables effectively. This document covers different table styles using the `tabular`, `tabularx`, `longtable`, `multirow`, and `booktabs` packages.

Basic Table Structure

Tables in LaTeX are created using the `tabular` environment:

```
\begin{tabular}{| l || c | p{0.3\textwidth} |}
    \hline
    Header 1 & Header 2 & Header 3 \\
    \hline\hline
    Cell 1 & Cell 2 & Cell 3 \\
    \hline
    Cell 4 & Cell 5 & Example text here. \\
    \hline
\end{tabular}
```

Using `table` Environment with Captions

To include captions and labels for referencing, wrap `tabular` inside a `table` environment:

```
\begin{table}[htbp]
    \centering
    \begin{tabular}{| l || c | p{0.3\textwidth} |}
        \hline
        Header 1 & Header 2 & Header 3 \\
        \hline\hline
        Cell 1 & Cell 2 & Cell 3 \\
        \hline
    \end{tabular}
    \caption{A Simple Table}
    \label{tab:simple}
\end{table}
```

You can reference this table using `\ref{tab:simple}` or `\autoref{tab:simple}`.

Custom Column Types

Custom column types using `array` package allow precise alignment:

```
\newcolumntype{C}{ >{\centering\arraybackslash} p }
\newcolumntype{R}{ >{\raggedleft\arraybackslash} p }
\newcolumntype{Y}{ >{\centering\arraybackslash} X }
```

`tabularx` for Dynamic Column Widths

The `tabularx` package allows dynamic width adjustments:

```
\begin{tabularx}{0.8\textwidth}{| l || c | X | Y |}
    \hline
    Header 1 & Header 2 & Header 3 & Header 4 \\
    \hline\hline
    Cell 1 & Cell 2 & Cell 3 & Cell 4 \\
    \hline
\end{tabularx}
```

Merging Rows and Columns

The `multirow` package enables merging rows and columns:

```
\begin{tabular}{ | *{5}{c |}}
    \hline
    \multicolumn{3}{|c|}{Merged} & Cell 1,4 & Cell 1,5 \\
    \hline
    \multirow{3}{*}{Grand Merge} & \multicolumn{2}{|c|}{Merged Rows} & Cell 3,5 & Cell 4,5 \
    \hline
\end{tabular}
```

Styling Tables with `booktabs`

The `booktabs` package provides professional styling:

```
\begin{table}[htbp]
    \centering
    \begin{tabular}{c c c}
        \toprule
        Header 1 & Header 2 & Header 3 \\
        \midrule
        Cell 1 & Cell 2 & Cell 3 \\
        \cmidrule{2-3}
        Cell 4 & Cell 5 & Cell 6 \\
        \bottomrule
    \end{tabular}
    \caption{A Styled Table}
    \label{tab:booktabs}
\end{table}
```

Handling Long Tables

The `longtable` package allows tables to span multiple pages:

```
\begin{longtable}[c]{| c | c |}
 \caption{Long table caption.\label{tab:long}}\\
 \hline
 \multicolumn{2}{|c|}{Begin of Table}\\
 \hline
 Something & something else\\
 \hline
 \endfirsthead

 \hline
 \multicolumn{2}{|c|}{Continuation of Table \ref{tab:long}}\\
 \hline
 Something & Not something else\\
 \hline
 \endhead

 \hline
 \multicolumn{2}{|c|}{Repeated foot} \\
 \hline
 \endfoot

 \hline
 \multicolumn{2}{|c|}{End of Table}\\
 \hline
 \endlastfoot

 Lots of lines & like this\\
 \end{longtable}
```

Conclusion

This document covered fundamental and advanced techniques for creating tables in LaTeX. Using these approaches, you can generate well-structured, readable, and professionally formatted tables in your documents.

LaTeX Detailed Tables Documentation

Introduction

Tables are a fundamental part of document preparation in LaTeX, enabling structured data representation. This document explains various table environments, column specifications, and additional enhancements using packages.

Required Packages

Before working with tables, ensure the following packages are included:

```
\usepackage{array}
\usepackage{tabularx}
\usepackage{multirow}
\usepackage{booktabs}
\usepackage{longtable}
```

Description of Packages:

- **array**: Enhances table column customization.
- **tabularx**: Adjusts column widths automatically.
- **multirow**: Allows merging rows.
- **booktabs**: Provides professional-looking tables.
- **longtable**: Supports tables spanning multiple pages.

Basic Table Structure

A simple table is created using the `tabular` environment:

```
\begin{tabular}{| l || c | p{0.3\textwidth} |}
    \hline
    Header 1 & Header 2 & Header 3 \\
    \hline \hline
    Cell 1 & Cell 2 & Cell 3 \\
    \hline
    Cell 4 & Cell 5 & Text spanning multiple lines. \\
    \hline
\end{tabular}
```

Column Specifiers:

- `l`: Left-aligned column.
- `c`: Center-aligned column.
- `r`: Right-aligned column.
- `p{width}`: Paragraph-style column with specified width.
- `|`: Draws vertical lines.
- `||`: Double vertical lines.

Adding Captions and Labels

Tables can include captions and labels for referencing:

```
\begin{table}[htbp]
    \centering
    \begin{tabular}{| l | c | p{0.3\textwidth} |}
```

```
        \hline
        Header 1 & Header 2 & Header 3 \\
        \hline
        Cell 1 & Cell 2 & Cell 3 \\
        \hline
    \end{tabular}
    \caption{A simple table}
    \label{tab:simple}
\end{table}
```

Reference the table in text using:

```
See Table \ref{tab:simple}.
```

Custom Column Types

Using `array`, define custom column types:

```
\newcolumntype{C}{>{\centering\arraybackslash} p}
\newcolumntype{R}{>{\raggedleft\arraybackslash} p}
```

- `C`: Centered paragraph column.
- `R`: Right-aligned paragraph column.

Example usage:

```
\begin{tabular}{| C{0.3\textwidth} |}
    \hline
    Centered text column \\
    \hline
\end{tabular}
```

Explanation of `\newcolumntype{C}{>{\centering\arraybackslash} p}` and `\newcolumntype{R}{>{\raggedleft\arraybackslash} p}`

In LaTeX, when creating tables, the `tabular` environment allows defining column alignments using predefined column types such as: - `l` for left-aligned columns, - `c` for centered columns, - `r` for right-aligned columns, - `p{width}` for paragraph-type columns with specific width.

However, LaTeX also provides the `array` package, which allows defining custom column types using `\newcolumntype`. This is useful when you want to apply a consistent formatting style across multiple tables.

Breaking Down `\newcolumntype{C}{>{\centering\arraybackslash} p}`

- `\newcolumntype{C}{...}`
  - This defines a new column type named `C` (capital C). You can now use `C{width}` in tables instead of `p{width}`.

- `{>{\centering\arraybackslash} p}`
  - `>` is used to modify the appearance of the following column type.
  - `\centering` ensures that the text in this column is **centered horizontally**.
  - `\arraybackslash` restores the standard behavior of `\\` inside the column.
  - `p` is the base column type, which allows **paragraph-style** text with a specified width.

Example Usage:

```
\begin{tabular}{|C{3cm}|C{5cm}|}
    \hline
    Column 1 & Column 2 \\
    \hline
    Some text & This text will be centered in both columns \\
    \hline
\end{tabular}
```

Here, `C{3cm}` means a **3 cm wide column with centered text**, and `C{5cm}` means a **5 cm wide column with centered text**.

****Breaking Down `\newcolumntype{R}{>{\raggedleft\arraybackslash} p}`**

- `\newcolumntype{R}{...}`
  - This defines a new column type named `R` (capital R). You can now use `R{width}` in tables instead of `p{width}`.
- `{>{\raggedleft\arraybackslash} p}`
  - `\raggedleft` ensures that text in this column is **right-aligned** instead of justified.
  - `\arraybackslash` restores the normal behavior of `\\`.
  - `p` means this is a **paragraph-style** column with a specified width.

Example Usage:

```
\begin{tabular}{|R{3cm}|R{5cm}|}
    \hline
    Column 1 & Column 2 \\
    \hline
    Some text & This text will be right-aligned in both columns \\
    \hline
\end{tabular}
```

Here, `R{3cm}` means a **3 cm wide column with right-aligned text**, and `R{5cm}` means a **5 cm wide column with right-aligned text**.

Why Use Custom Column Types?

1. **Improves Readability**: Instead of writing `>{\centering}p{width}` every time, you can just use `C{width}`.
2. **Maintains Consistency**: If you use `C` or `R` throughout multiple tables, formatting stays uniform.
3. **Saves Time**: You define the column type once and reuse it across the document.

Would you like me to explain another part in more detail?

Advanced Tables

Multirow and Multicolumn Cells

Merge columns and rows using `multirow` and `multicolumn`:

```
\begin{tabular}{|c|c|c|c|}
    \hline
    \multicolumn{2}{|c|}{Merged Columns} & Cell 3 & Cell 4 \\
    \hline
    Cell 1 & \multirow{2}{*}{Merged Rows} & Cell 3 & Cell 4 \\
    \cline{1-1} \cline{3-4}
    Cell 2 &  & Cell 3 & Cell 4 \\
    \hline
\end{tabular}
```

TabularX for Auto-Adjusting Width

`tabularx` allows automatic column width adjustment:

```
\begin{tabularx}{0.8\textwidth}{|X|X|}
    \hline
    Column 1 & Column 2 \\
    \hline
    Text in the first column & Text in the second column \\
    \hline
\end{tabularx}
```

- `X`: Automatically expanding column width.

Booktabs for Professional Tables

For cleaner formatting:

```
\begin{tabular}{c c c}
    \toprule
    Header 1 & Header 2 & Header 3 \\
    \midrule
    Data 1 & Data 2 & Data 3 \\
```

```
    \bottomrule
\end{tabular}
```

- `\toprule`: Top line.
- `\midrule`: Middle line.
- `\bottomrule`: Bottom line.

Long Tables Spanning Multiple Pages

For large tables:

```
\begin{longtable}{|c|c|}
    \caption{A long table}\
    \hline
    Header 1 & Header 2 \\
    \hline
    \endfirsthead

    \hline
    Header 1 & Header 2 \\
    \hline
    \endhead

    Data & More Data \\
    \hline
    \endfoot

    \hline
    End of table \\
    \hline
    \endlastfoot
\end{longtable}
```

Additional Information

Explanation of Table Formatting Commands

If the following commands were **uncommented**, they would modify the appearance of all tables in your LaTeX document:

```
\setlength{\arrayrulewidth}{1.2pt}
\setlength{\tabcolsep}{15pt}
\renewcommand{\arraystretch}{0.5}
```

---

1. `\setlength{\arrayrulewidth}{1.2pt}`

What it does:

This command **sets the thickness of the table borders (horizontal and vertical lines)** in all `tabular` environments.

- The default thickness is **0.4pt** (points).
- By setting it to `1.2pt`, table borders become three times thicker.

Example (before and after)

Before (default 0.4pt)

```
\setlength{\arrayrulewidth}{0.4pt}  % Default
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

The table has thin borders.

After (`1.2pt`)

```
\setlength{\arrayrulewidth}{1.2pt}  % Thicker lines
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Now, the table lines are **bold and more visible**.

---

2. `\setlength{\tabcolsep}{15pt}`

What it does:

This command **adjusts the spacing (padding) between columns** in all tables.

- The default column separation is **6pt**.
- Increasing it to `15pt` **adds more space** between columns, making tables look less crowded.

Example (before and after)

Before (default 6pt)

```
\setlength{\tabcolsep}{6pt}  % Default
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Columns are **closer together**.

After (15pt)

```
\setlength{\tabcolsep}{15pt}  % Wider column spacing
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Columns **now have more space** between them.

When to Use?

- If your table looks too cramped, increase `\tabcolsep`.
- If you need a compact table, **reduce** this value.

---

3. `\renewcommand{\arraystretch}{0.5}`

What it does:

This **controls the vertical spacing (height) between rows** in all tables.

- The default value is `1.0` (normal height).
- Setting it to `0.5` **shrinks row height**, making tables more compact.

Example (before and after)

Before (default `1.0`)

```
\renewcommand{\arraystretch}{1.0}  % Default
\begin{tabular}{|c|c|}
    \hline
```

```
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Normal row height.

After (0.5)

```
\renewcommand{\arraystretch}{0.5}  % Smaller row spacing
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Rows are **tightly packed**.

After (1.5)

```
\renewcommand{\arraystretch}{1.5}  % Larger row spacing
\begin{tabular}{|c|c|}
    \hline
    A & B \\
    \hline
    1 & 2 \\
    \hline
\end{tabular}
```

Rows are **more spaced out**, making the table easier to read.

When to Use?

- Use a **lower value** (`<1.0`) to **compress tables** when space is limited.
- Use a **higher value** (`>1.0`) for **better readability**.

Final Thoughts

Command	Purpose	Effect
`\setlength{\arrayrulewidth}{1.2pt}`	Controls **table border thickness**	Thicker table lines
`\setlength{\tabcolsep}{15pt}`	Controls **column spacing**	More space between columns

Command	Purpose	Effect
`\renewcommand{\arraystretch}{1.5}`	**Control row spacing**	Tighter or looser row height

Conclusion

LaTeX provides a variety of ways to create structured tables, ranging from simple grids to advanced multi-page tables. Using the appropriate packages and formatting techniques ensures professional and readable tables.