OFFICE & PRODUCTIVITY SOFTWARE

LibreOffice

Writer vs. Word · Page Styles & Mail Merge

Calc vs. Excel · Impress vs. PowerPoint

File Format Interoperability in Practice

Capstone: Rebuilding a Complete Project in LibreOffice

Philip Osztromok

Generated with Claude

Table of Contents

LibreOffice — 10 Chapters

This course assumes you've already been through Excel Fundamentals/Advanced and Word & PowerPoint on this site. Rather than re-teaching styles, formulas, or slide layouts from zero, every chapter here says, in effect, "you already know this concept — here's specifically what's different in LibreOffice." That framing starts now, with what LibreOffice actually is and the one practical issue that runs through nearly every chapter that follows: file format interoperability.

What LibreOffice Actually Is

LibreOffice is a free, open-source office suite developed by **The Document Foundation**, running on Windows, macOS, and Linux. It bundles several applications, three of which map directly onto apps this site's courses already cover: **Writer** (documents), **Calc** (spreadsheets), and **Impress** (presentations). LibreOffice also includes Draw, Base, and Math — genuinely useful apps in their own right, but outside this course's scope, since there's no directly matching course on this site to compare them against.

Free and Open-Source vs. Proprietary: What Actually Differs

LibreOffice costs nothing — no subscription, no per-seat license — and its source code is publicly available and modifiable by anyone, governed by a non-profit foundation rather than a single company's own product roadmap. For a typical day-to-day user, though, this distinction matters less than it might sound: the actual daily experience of writing a document or building a spreadsheet is broadly similar either way. The differences that genuinely matter in practice show up somewhere much more concrete — file formats, and a handful of feature gaps — which is exactly what the rest of this course focuses on.

The Suite Model: Writer, Calc, Impress

LIBREOFFICE APP	MICROSOFT OFFICE EQUIVALENT	COVERED BY
Writer	Word	Word & PowerPoint Chapters 2-7
Calc	Excel	The full Excel Fundamentals/Advanced track
Impress	PowerPoint	Word & PowerPoint Chapters 8-11

The Open Document Format (ODF)

LibreOffice's own native file format is **ODF** (Open Document Format) — `.odt` for Writer documents, `.ods` for Calc spreadsheets, `.odp` for Impress presentations. ODF is an open, ISO-standardized format, genuinely different from Microsoft's own OOXML-based `.docx`/`.xlsx`/`.pptx`. LibreOffice can open and save Microsoft's formats directly too — but ODF, not a Microsoft format, is what LibreOffice defaults to and considers its own native home.

Why File Format Interoperability Matters

Most of the world still works in Microsoft's own file formats, which means using LibreOffice day to day usually means constantly opening and saving files in a format that *isn't* its native one. Every round-trip between ODF and a Microsoft format carries a real, if usually small, risk that something doesn't translate perfectly — a specific formatting detail, a complex field, a chart type that has no exact equivalent on the other side. This isn't a flaw unique to LibreOffice specifically; it's an unavoidable consequence of two genuinely different file formats trying to represent each other's features. Chapter 9 covers the concrete specifics of what typically survives cleanly and what commonly doesn't.

The "Same Job, Different App" Framing This Course Uses

Every chapter from here on will explicitly name which chapter of the Excel or Word & PowerPoint courses covers the equivalent concept, then focus disproportionately on what's genuinely *different* about LibreOffice's own approach — not repeat material you've already learned. Some of those differences are trivial (a menu in a different place); some are genuinely different mental models worth understanding properly (Writer's Page Styles, covered in Chapter 3, being the clearest example).

	MICROSOFT OFFICE	LIBREOFFICE
Cost	Subscription or one-time license	Free
Source model	Proprietary, closed source	Open source, community/foundation-governed
Native document format	.docx / .xlsx / .pptx (OOXML-based)	.odt / .ods / .odp (ODF, ISO-standardized)
Can open the other suite's files	Limited native ODF support	Yes, directly — but not always losslessly

THIS COURSE LEANS ON THE EXCEL AND WORD & POWERPOINT COURSES DIRECTLY

Every chapter ahead references specific chapters from those two completed courses by number — being genuinely comfortable with Styles (Word & PowerPoint Ch.2), section breaks (Ch.3), Excel's own PivotTables (Excel Fundamentals Ch.9), and the Slide Master (Word & PowerPoint Ch.8) will make this course click considerably faster, since this course assumes that grounding rather than re-establishing it.

A SAVE-FORMAT PROMPT IS EASY TO CLICK THROUGH ON AUTOPILOT

Saving a file that was originally opened in a Microsoft format prompts a dialog asking whether to keep that format ("Use Word 2007-365!") or switch to ODF ("Use ODF Format!"). Clicking through this prompt out of habit — repeatedly choosing whichever button muscle memory reaches for — can silently convert a file's actual format without genuinely meaning to. This matters in particular if the file needs to go back to a colleague who only has Microsoft Office: an accidentally-converted `.odt` file may not open cleanly for them at all. Always read that prompt deliberately rather than dismissing it automatically.

Hands-On Exercises

EXERCISE 1

Open LibreOffice Writer and create a new blank document. Save it once using the default Save command. Without changing any settings, note which file extension LibreOffice chooses by default, and explain why that specific extension is the one it picked.

View solution

EXERCISE 2

A colleague sends you a .xlsx file. You open it in LibreOffice Calc, make a small edit, and press Save. Describe exactly what dialog you'd expect to see, what the two main choices mean, and which one you should pick if this file needs to go back to that same colleague afterward.

View solution

EXERCISE 3

Match each LibreOffice application — Writer, Calc, Impress — to its Microsoft Office equivalent, and to the specific completed course/chapter range on this site that already taught the underlying concepts for that equivalent app.

View solution

CHAPTER 1 QUICK REFERENCE

- **Writer / Calc / Impress** — LibreOffice's equivalents of Word / Excel / PowerPoint
- **ODF** (.odt/.ods/.odp) — LibreOffice's own native, ISO-standardized format, distinct from Microsoft's OOXML-based formats
- LibreOffice can open/save Microsoft formats directly, but they aren't its native format — round-trips carry a real risk of small translation issues (Chapter 9)
- This course names the equivalent Excel/Word & PowerPoint chapter for each concept, then focuses on what's genuinely different
- Read the save-format prompt deliberately — don't click through it on autopilot when a file needs to stay in a specific format

Word & PowerPoint Chapter 2 established that a Style is a named, reusable formatting definition — modify it once, and every paragraph or text run tagged with it updates together, everywhere. That concept doesn't change one bit in Writer. What changes is entirely where you go to use it.

The Same Concept: Paragraph and Character Styles

Writer has the identical two-level style system Word & PowerPoint Chapter 2 introduced: a **paragraph style** applies to an entire paragraph at once (font, size, spacing, alignment, all bundled together); a **character style** applies only to a selected run of text within a paragraph, leaving everything else about that paragraph untouched. If you understood that distinction in Word, you already understand it here — nothing about the underlying idea is different.

The Styles Panel

F11 (or Styles → Manage Styles) opens the **Styles panel** — Writer's equivalent of Word's Ribbon-based Styles gallery. It's organised into several tabs: Paragraph Styles, Character Styles, Frame Styles, Page Styles, and List Styles. Word exposes mainly paragraph and character styles directly through its own gallery; Writer surfaces genuinely more categories in one place, including Page Styles — the subject of Chapter 3.

Applying a Style

Click into a paragraph (no need to select the whole thing), then double-click a style in the Styles panel to apply it — functionally identical to clicking a style in Word's own gallery. The style dropdown in Writer's own toolbar offers the same action even more directly, without needing the full panel open at all.

The Navigator: Writer's Equivalent of the Navigation Pane

F5 (or View → Navigator) opens a categorised, expandable tree of the **entire document's structure** — not just headings, the way Word's Navigation Pane primarily shows, but also tables, images, bookmarks, frames, and more, each in its own category. Double-clicking any item in the tree jumps directly to it, the same core convenience as Word's Navigation Pane, just covering a genuinely broader set of document elements at once.

Modifying a Style: The Same Central-Update Behaviour

Right-click any style in the Styles panel and choose **Modify** to change its definition — the update applies immediately to every paragraph or text run using that style throughout the document, exactly like modifying a style in Word. This is the same "define once, apply everywhere, update centrally" idea Word & PowerPoint Chapter 2 built its entire lesson around; nothing here is a new concept to learn, only a new panel to find it in.

CONCEPT	WORD	WRITER
Where styles live	Styles gallery, on the Home Ribbon	Styles panel (F11)
Style categories exposed	Mainly paragraph and character	Paragraph, Character, Frame, Page, List — all in one panel
Structural navigation	Navigation Pane — headings-focused	Navigator (F5) — headings, tables, images, bookmarks, and more
Modifying a style's definition	Right-click → Modify	Right-click → Modify

IF YOU UNDERSTAND WORD & POWERPOINT CHAPTER 2, YOU ALREADY UNDERSTAND MOST OF THIS CHAPTER

The genuine delta between Word and Writer here is small: a different panel (F11 vs. the Ribbon gallery), a broader Navigator instead of a headings-focused Navigation Pane, and a couple of extra style categories exposed in the same panel. The underlying "why" — named, reusable, centrally-updatable formatting — carries over completely unchanged.

ROUND-TRIPPING A DOCUMENT BETWEEN WORD AND WRITER CAN PRODUCE DUPLICATE OR MISMATCHED STYLE NAMES

A document originally styled in Word, then opened and further edited in Writer, then saved back to .docx, can occasionally end up with near-identical style names (e.g. two subtly different "Heading 1" definitions) or a style Writer created during editing that Word doesn't recognise cleanly. This connects directly to Chapter 1's own interoperability warning, and Chapter 9 covers the specifics — for now, it's worth knowing that style names themselves aren't perfectly immune to translation issues across a format round-trip, even though the underlying style CONCEPT transfers cleanly.

Hands-On Exercises

EXERCISE 1

In a new Writer document, type three section titles with body paragraphs beneath each. Apply the Heading 1 paragraph style to all three titles via the Styles panel (F11). Open the Navigator (F5) and confirm all three appear. Then modify the Heading 1 style's colour and describe what happens to all three titles.

View solution

EXERCISE 2

Add a table and an image to the same document from Exercise 1. Open the Navigator again and describe what additional categories now appear that weren't there before, and contrast this with what Word's own Navigation Pane would show for the same document.

View solution

EXERCISE 3

Within one body paragraph, select a single word and apply the built-in "Strong Emphasis" character style. Explain what changed about that one word versus the rest of the paragraph, and identify the exact Word & PowerPoint chapter and concept this directly mirrors.

View solution

CHAPTER 2 QUICK REFERENCE

- **Paragraph style** vs. **character style** — identical concept to Word & PowerPoint Chapter 2, no changes to the underlying idea
- **F11** — the Styles panel, Writer's equivalent of Word's Styles gallery, with more categories in one place
- **F5** — the Navigator, a broader structural tree than Word's headings-focused Navigation Pane
- Modifying a style updates every paragraph/run using it, exactly like Word
- A Word-to-Writer-and-back round trip can occasionally produce mismatched or duplicated style names — see Chapter 9

Chapter 2's differences from Word were mostly cosmetic — the same concepts, reached through a different panel. This chapter is genuinely different. Writer has no concept of a "section" at all. Everything Word & PowerPoint Chapter 3 taught you to solve with section breaks — a landscape page mid-document, a different header partway through — Writer solves with an entirely different mechanism: **Page Styles**.

Recap: How Word Solves This

Word & PowerPoint Chapter 3 established that margins, orientation, and headers are properties of a **section** — a region of the document bounded by section breaks. To give one page a different orientation, you insert two section breaks around it, then apply the new orientation to that isolated section specifically.

Writer's Different Approach: Page Styles

In Writer, **every single page already belongs to a named Page Style** — most commonly "Default Page Style," at first. Margins, orientation, and header/footer content are all properties of that Page Style itself, not of a section boundary. To change formatting mid-document, you don't insert a break that divides the document — you insert a manual page break that explicitly **switches which Page Style** applies from that point onward.

Inserting a Page Break with a Page Style Change

Insert → **More Breaks** → **Insert Page Break** opens a dialog offering, in one step: the break Type (Page break), a **Style** dropdown choosing which Page Style the new page should use, and an optional page-number restart. Word requires a section break first, and then separately toggling orientation and separately opening Format Page Numbers; Writer bundles all three decisions into this one dialog at the moment the break itself is inserted.

Built-In Page Styles You'll Actually Use

- **Default Page Style** — the ordinary, standard page.
- **Landscape** — already predefined with landscape orientation; switching to it is all that's needed, with no separate orientation toggle required afterward.
- **First Page** — designed for a document's opening page, commonly with no header defined at all.

Creating or Modifying a Custom Page Style

Page Styles live in the exact same **Styles panel** (F11) Chapter 2 introduced — switch to its **Page Styles** tab, right-click, and choose New or Modify. Page Styles genuinely are just one more style category in the same system Chapter 2 already covered, not a separate, unrelated feature — they simply govern page-level properties instead of paragraph-level ones.

Headers and Footers Are Also Tied to the Page Style

This is where Writer's model pays off structurally: because a Page Style owns its own header and footer definition directly, switching to a different Page Style mid-document automatically gives that region its own different header — with **no separate "Link to Previous" toggle needed at all**. Word & PowerPoint Chapter 5's entire Link to Previous mechanism exists to solve a problem that, in Writer, is already solved by the simple fact that a header belongs to whichever Page Style is currently active.

TASK	WORD (CHAPTERS 3 & 5)	WRITER
Creating a new formatting region	Insert a section break	Insert a page break, choosing a new Page Style
Where margins/orientation live	The section	The Page Style
Giving a region its own header	Turn off "Link to Previous"	Automatic — the header belongs to that Page Style already
Restarting page numbers	Format Page Numbers → Start at	The same Insert Page Break dialog's own page-number option

THIS ISN'T A NEW UI SURFACE — IT'S CHAPTER 2'S OWN STYLES PANEL AGAIN

Page Styles aren't a separate feature bolted onto Writer alongside its style system — they're one more tab in the exact same Styles panel (F11) Chapter 2 already introduced. If Chapter 2's "define once, apply everywhere, update centrally" idea made sense for paragraphs, this chapter is that same idea applied one level up, to whole pages.

THERE'S NO "CLOSING" A PAGE STYLE REGION, AND EDITING A HEADER CAN REACH FURTHER THAN EXPECTED

Because Writer has no section concept to delete or merge, returning to an earlier Page Style later in the document requires inserting *another* manual page break that explicitly switches back — there's no equivalent of deleting a Word section break to collapse two regions into one. Separately, and genuinely important: editing a Page Style's header changes it for **every page anywhere in the document currently using that Page Style** — not just "the rest of the document from here on," the way turning off Link to Previous in Word affects only what follows. If "Default Page Style" is used both before and after a Landscape interruption, editing its header once changes both regions at the same time, since they share the exact same named style.

Hands-On Exercises

EXERCISE 1

In a Writer document with several portrait pages, place your cursor where you want a wide table to begin. Use Insert > More Breaks > Insert Page Break to switch to the Landscape page style for that one page, then insert another page break switching back to Default Page Style immediately after. Describe exactly what settings you chose in each dialog and why two separate page breaks were needed.

View solution

EXERCISE 2

A document has pages 1-3 and pages 5-6 both using "Default Page Style" (page 4 uses "Landscape" in between). You edit the header while your cursor is on page 6. Explain exactly what happens to the header on pages 1-3, and why this differs from how a similar edit would behave in Word with Link to Previous turned off on a later section.

View solution

EXERCISE 3

A colleague coming from Word asks why Writer's Insert Page Break dialog has a "Style" dropdown at all — in Word, they say, a page break "just makes a new page, nothing more." Explain why that dropdown exists and what it lets Writer accomplish in one step that Word cannot do with a plain page break alone.

View solution

CHAPTER 3 QUICK REFERENCE

- Writer has **no section concept** — every page belongs to a named **Page Style** instead
- **Insert → More Breaks → Insert Page Break** — choose the new Page Style and an optional page-number restart, all in one dialog
- **Landscape** Page Style already has landscape orientation predefined; **First Page** is suited to a document's opening page
- Page Styles live in the same **Styles panel** (F11) as paragraph/character styles, just their own tab
- A Page Style's header belongs to *every* page using that style, not just "from here forward" — editing it can reach both before and after a switched-out region
- Returning to an earlier Page Style requires another explicit page break — there's no equivalent of merging sections back together

Word & PowerPoint Chapter 7 established the underlying job: one template, one data source, many personalized documents. That job hasn't changed. What's genuinely different here is the flow — Writer walks you through it with a guided, step-by-step **Mail Merge Wizard**, rather than the more free-form, Ribbon-driven process Word uses.

The Same Job, a Different Flow: The Mail Merge Wizard

Tools → **Mail Merge Wizard** opens an 8-step guided sequence: select a starting document, choose the document type (letter or email), insert an address block, create a salutation, adjust the layout, edit merge fields directly if needed, personalize and preview the documents, then complete the merge. Word offers the same overall set of decisions, just made in whichever order you choose from its Mailings tab rather than a fixed guided sequence.

Choosing the Data Source

Within the wizard, **Select Address List** connects to your data — a Calc spreadsheet (.ods or .xlsx), or an external database registered with LibreOffice. This is genuinely broader than Word's own merge, which is built primarily around file-based sources; Writer's merge is designed from the ground up to work against real database connections too, not just spreadsheet files.

The Registered Data Sources Concept

Writer's merge mechanism is built around **registering** a data source with LibreOffice first — giving it a name LibreOffice remembers (via Tools → Options → LibreOffice Base → Databases, or File → New → Database) — rather than browsing fresh to a file every single time, the way Word's Select Recipients → Use an Existing List does. Once registered, that same named data source can be reused across multiple merge documents without ever re-browsing to the underlying file again.

Address Block and Salutation: Built-In Smart Matching

Steps 3 and 4 of the wizard attempt to automatically match your data source's own column names to the fields an address block or salutation expects — a "Match Fields"-style step conceptually identical to Word & PowerPoint Chapter 7's own Match Fields dialog, and driven by the exact same underlying concern: column names that don't match exactly need to be explicitly mapped, or the merge silently produces blank or mismatched results.

Inserting Additional Merge Fields Manually

Insert → Field → More Fields → Database tab (also reachable directly from within the wizard) lets you place any specific column's value anywhere in the document — the same underlying action as Word's Insert Merge Field, just reached through a general-purpose Field dialog rather than one dedicated Ribbon button.

Personalizing and Completing the Merge

The wizard's later steps preview each record's actual data filled into the document — the same purpose as Word's own Preview Results — before finishing by printing directly, saving the merge as one combined document, or emailing each personalized copy: the same three completion options Word & PowerPoint Chapter 7's Finish & Merge offers.

ASPECT	WORD	WRITER
Starting flow	Free-form, via the Mailings tab	A fixed, guided 8-step Wizard
Connecting a data source	Browse to a file fresh each time	Register the source once by name, reuse it across merges
Field name matching	Match Fields dialog	Built into the wizard's address block/salutation steps
Data source types supported	Mostly file-based (Excel, etc.)	Spreadsheets and real registered database connections

THE SAME CLEAN-SPREADSHEET LESSON FROM EXCEL FUNDAMENTALS STILL APPLIES

The exact Excel Table used for Word & PowerPoint's own capstone Mail Merge — clear column headers, one record per row, no blank rows scattered through it — could, in principle, serve as this Writer merge's own data source too, since Calc opens .xlsx files directly. Excel Fundamentals Chapter 7's "a well-structured data source matters" lesson travels across both apps and both suites unchanged; only the mechanism connecting to it looks different.

A REGISTERED DATA SOURCE IS TIED TO A FILE PATH, AND THAT PATH CAN GO STALE

Registering a data source links it to the exact file location it pointed at when registered. If that file is later moved, renamed, or deleted, the registration doesn't update itself — the next merge attempt using that registered name simply fails to resolve it, requiring the source to be re-registered against its new location. This is a genuinely different failure mode from Word's simpler "browse to the file fresh each time" approach, which has no equivalent stale-registration problem, precisely because it never remembers a file's location between merges in the first place.

Hands-On Exercises

EXERCISE 1

Create a small Calc spreadsheet with columns Name and Amount, 3 rows of sample data. Register it as a data source with LibreOffice, then run through the Mail Merge Wizard in Writer to produce a personalized letter per row. Describe which wizard step connects to your registered source, and which step lets you preview one row's actual data before finishing.

View solution

EXERCISE 2

Six months after Exercise 1, the original spreadsheet file is moved to a different folder and renamed. You open the same Writer template and try to run the merge again using the same registered data source name. Describe what happens and what step is needed to fix it.

View solution

EXERCISE 3

Your spreadsheet's columns are named CustName and AmtDue, but the wizard's address block step expects fields named Name and Amount. Explain what would happen if you proceeded without addressing this, and identify the exact Word & PowerPoint Chapter 7 concept this situation directly mirrors.

View solution

CHAPTER 4 QUICK REFERENCE

- **Tools → Mail Merge Wizard** — an 8-step guided flow, same underlying job as Word's Mailings tab
- Data sources are **registered** by name with LibreOffice, then reused across merges without re-browsing
- Writer's merge supports both spreadsheets and real database connections, more broadly than Word's mostly file-based sources
- Field-name matching in the wizard is the same exact-match concern as Word & PowerPoint Chapter 7's own Match Fields
- A registered data source's file path can go stale if the file is moved or renamed, requiring re-registration

This one chapter stands in for the entire 20-chapter Excel Fundamentals/Advanced track. Nearly everything that track taught — the grid model, cell references, $ locking, functions, lookups, dynamic arrays — transfers to Calc essentially unchanged. This chapter isn't re-teaching any of that; it's isolating the genuine syntax-level differences actually worth knowing.

The Same Underlying Model

Calc is a grid of independently addressable cells, exactly as Excel Fundamentals Chapter 1 described. Cell addresses (`B7`), ranges (`A1:A10`), and relative/absolute `$` locking (`$A$1` , `$A1` , `A$1`) all work identically, with the same syntax, the same meaning, and the same behaviour when filled or copied.

The Argument Separator: Comma vs. Semicolon

The one genuinely concrete syntax difference worth knowing up front: depending on your system's regional/locale settings, Calc may expect a **semicolon** rather than a comma between function arguments:

```
Excel, and Calc under a comma-separator locale:
=SUM(A1,A2,A3)
=IF(B2>=50,"Pass","Fail")

Calc under a semicolon-separator locale (equally valid, same result):
=SUM(A1;A2;A3)
=IF(B2>=50;"Pass";"Fail")
```

This isn't a fixed, universal rule — it's inherited from LibreOffice's own regional settings (**Tools** → **Options** → **Calc** → **Formula**), so it can genuinely differ between installations and users. If a formula typed with commas produces an error, checking that setting is the first thing worth doing.

Occasional Function Name Differences

The overwhelming majority of function names are identical between the two applications — `SUM`, `AVERAGE`, `IF`, `VLOOKUP`, `XLOOKUP`, `IFERROR` all exist under the same names in current Calc. A small number of edge cases exist around newer or less common functions — particularly anything introduced very recently in one application that hasn't yet been matched in the other. The safe habit, rather than relying purely on memory for an unfamiliar or brand-new function, is checking the Function Wizard before assuming a name transfers unchanged.

The Function Wizard

Insert → **Function** (or the *fx* icon on the formula bar) opens Calc's own guided function-building dialog — browsable by category, showing each function's exact name and expected argument order. This is the equivalent of Excel's own Insert Function dialog, and the most reliable way to confirm a function genuinely exists under the name you expect before typing it from memory.

Formula Syntax That Stays Identical

Worth stating plainly, since this chapter has focused on differences: cell references, arithmetic operators (`+ - * / ^`), string concatenation with `&`, and comparison operators (`= <> > < >= <=`) are all completely unchanged. The overwhelming majority of everything the Excel track taught applies to Calc with zero translation needed at all.

ELEMENT	EXCEL	CALC
Cell references, $ locking	A1, A1, $A1, A$1	Identical
Argument separator	Comma	Comma or semicolon, depending on locale settings
Common function names	SUM, IF, VLOOKUP, XLOOKUP, IFERROR	Identical, in almost every case
Function-lookup tool	Insert Function dialog	Function Wizard (Insert → Function)

THIS IS USUALLY HANDLED AUTOMATICALLY, NOT SOMETHING YOU NEED TO FIX BY HAND

A formula typed with commas, opened in a Calc installation whose locale expects semicolons, is typically translated automatically the moment the file opens or saves — this isn't normally something a user needs to manually rewrite. It's still worth knowing the underlying reason exists, per Chapter 1's own interoperability point, if a formula ever looks unexpectedly different after a round trip between the two formats.

DON'T ASSUME EVERY FUNCTION NAME TRANSFERS FROM MEMORY — CHECK WHEN UNSURE

Most function names genuinely are identical, but assuming this holds universally, without checking, is the actual risk — particularly for a function introduced very recently in one application that the other hasn't caught up to yet. The Function Wizard costs one extra click and removes the guesswork entirely; relying on memory for an unfamiliar function name is the habit actually worth avoiding here.

Hands-On Exercises

EXERCISE 1

Type =SUM(A1,A2,A3) into a Calc cell. If it produces an error, check Tools > Options > Calc > Formula to find the actual separator your installation expects, and rewrite the formula using that separator instead. Explain why the exact same underlying formula might need a different separator character on a different computer.

View solution

EXERCISE 2

Open the Function Wizard (Insert > Function) and locate XLOOKUP. Confirm its argument order matches what Excel Fundamentals Chapter 5 taught. Explain why checking the Function Wizard is a better habit than typing a function purely from memory, even for a function you're confident about.

View solution

EXERCISE 3

A colleague emails you an Excel file with the formula =IF(B2>=50,"Pass","Fail") in a cell. You open it in Calc on a machine configured with a semicolon argument separator. Describe what you'd expect to see in that cell's formula bar, and explain why you likely don't need to manually fix anything.

View solution

CHAPTER 5 QUICK REFERENCE

- Calc's grid, references, and $ locking are **identical** to Excel — no translation needed
- The **argument separator** (comma vs. semicolon) depends on Tools → Options → Calc → Formula, not a fixed universal rule
- Most function names transfer **unchanged**; check the **Function Wizard** (Insert → Function) for anything unfamiliar or very new
- Arithmetic operators, `&` concatenation, and comparison operators are all identical
- Separator mismatches after a file round-trip are usually handled automatically, not something to manually fix

Excel Fundamentals Chapter 9 built a PivotTable by dragging fields into a live, docked panel and watching the result update on the sheet as you went. Calc solves exactly the same problem — summarizing raw data without writing a formula — but through a genuinely different interaction model: a static dialog you configure fully before anything appears.

The Same Job: Summarizing Data Without Writing a Formula

The underlying need hasn't changed from Excel Fundamentals Chapter 9: raw transaction data, one row per record, and a question like "what's the total by category?" that a Pivot Table answers by grouping and aggregating automatically, rather than requiring a separate formula per group.

Building a Pivot Table

Data → Pivot Table → Insert opens a **Layout dialog** with four boxes — Page Fields, Column Fields, Row Fields, and Data Fields — roughly mapping onto Excel's own Filters, Columns, Rows, and Values. Field names are dragged from an "Available Fields" list directly into these boxes, entirely within this one static dialog.

The Field-Assignment Dialog vs. Excel's Live Panel

This is the genuine interaction difference: Excel lets you drag fields into a persistent side panel and see the actual PivotTable update live on the sheet with every change. Calc instead asks you to configure the **entire layout inside one dialog first** — Page Fields, Column Fields, Row Fields, and Data Fields all placed before clicking OK — and only then does the finished Pivot Table appear on the sheet. It's a "configure everything, then generate" flow, rather than Excel's own "drag and immediately see the result" flow.

Changing the Aggregation

Double-clicking a field already placed in the Data Fields box (still inside the Layout dialog) opens a small "Data Field" dialog offering the same aggregation choices Excel's own Value Field Settings does — Sum, Average, Count, Max, Min, and more — just reached from within the layout-configuration dialog rather than by clicking a field already sitting in a live Values area.

Refreshing

Right-click the Pivot Table and choose **Refresh** (or Data → Refresh Range/Pivot Table). This requirement is completely identical to Excel Fundamentals Chapter 9's own rule: the source data changing does **not** automatically update the Pivot Table — a manual refresh is required every time, in both applications, with no difference in behaviour here at all.

Editing an Existing Pivot Table's Layout

Right-click the Pivot Table and choose **Edit Layout** to reopen the same static dialog used to build it originally. Rearranging which fields sit in Rows versus Columns means going back into that dialog again — a genuinely different workflow from Excel, where a field can simply be dragged from one live area of the Fields panel directly to another, on the already-built PivotTable, with no separate dialog needed at all.

ASPECT	EXCEL (FUNDAMENTALS CH.9)	CALC
Building the layout	Live, docked Fields panel — see results as you drag	A static Layout dialog — configure fully, then click OK
Changing the aggregation	Value Field Settings, on a field already in Values	Double-click a Data Field inside the same Layout dialog
Refreshing after source data changes	Required manually	Required manually — identical behaviour, no difference
Rearranging fields afterward	Drag directly on the live PivotTable	Re-open the Layout dialog (Edit Layout)

THE FOUR ROLES ARE IDENTICAL — ONLY HOW YOU ASSIGN FIELDS TO THEM DIFFERS

Excel Fundamentals Chapter 9's real lesson wasn't about drag-and-drop mechanics — it was about recognising that any summary needs something to group rows by, optionally something to group columns by, something being aggregated, and optionally something filtering the whole thing. Calc's Layout dialog asks for exactly those same four roles, just via typed-in dialog boxes instead of a live panel. Understanding *why* those four roles exist, from Excel, is what actually transfers — the specific dialog is just a different way of answering the same four questions.

THE SAME NUMERIC-AS-TEXT GOTCHA APPLIES, AND MISTAKES ARE HARDER TO CATCH MID-CONFIGURATION

Excel Fundamentals Chapter 9's warning about a numeric-looking field stored as text defaulting to Count instead of Sum applies identically here — it isn't a new Calc-specific problem, just the same underlying issue (a column not being genuinely numeric) surfacing in a different tool. Separately: because Calc's Layout dialog requires the entire configuration to be finished before anything appears on the sheet, a field placed in the wrong box isn't visually obvious the way it might become quickly apparent in Excel's own live-updating panel — double-check every field's placement in the dialog itself before clicking OK, rather than expecting to immediately spot a mistake the way Excel's live preview would likely reveal one sooner.

Hands-On Exercises

EXERCISE 1

Given a small Calc data set with columns Region, Category, and Sales, build a Pivot Table with Region in Row Fields, Category in Column Fields, and Sales in Data Fields. Describe exactly which dialog you configured this in, and what you had to do differently compared to how Excel Fundamentals Chapter 9 described building the same layout.

View solution

EXERCISE 2

After building the Pivot Table from Exercise 1, you add 10 new rows to the underlying source data. Describe exactly what you need to do for the Pivot Table to reflect the new data, and explain why this requirement is not actually a LibreOffice-specific quirk.

View solution

EXERCISE 3

Your Sales column was accidentally imported as text rather than genuine numbers. You build a Pivot Table with Sales in Data Fields and it shows "Count" instead of "Sum," with numbers that clearly look like row counts rather than totals. Explain what's actually wrong and identify the exact Excel Fundamentals chapter and concept this mirrors.

View solution

CHAPTER 6 QUICK REFERENCE

- **Data → Pivot Table → Insert** — a static Layout dialog with Page/Column/Row/Data Fields boxes
- Configure the **entire layout first**, then click OK — no live preview while building, unlike Excel's docked panel
- Double-click a Data Field inside the dialog to change its aggregation (Sum/Average/Count/etc.)
- **Refresh** is required manually after source data changes — identical to Excel, not a LibreOffice-specific requirement
- **Edit Layout** reopens the same dialog to rearrange fields — no direct drag-to-rearrange on the finished Pivot Table
- The numeric-stored-as-text → Count-instead-of-Sum gotcha applies identically to Calc

Word & PowerPoint Chapter 8 established the core idea: define a slide's shared design once, and every slide built from it updates together. That idea is completely unchanged in Impress. What's worth understanding carefully here is that Impress organizes the "define once" system slightly differently than PowerPoint does — not wrong, just structured with a different split between two panels rather than one strict two-level hierarchy.

The Same Concept: Define Once, Apply Everywhere

Editing a shared design definition once and having every slide using it update automatically — the exact idea Word & PowerPoint Chapter 8 built its entire lesson around — works identically in Impress. The difference in this chapter is entirely about where that shared definition lives and how it's organized, not whether the underlying behaviour exists.

Master Slides in Impress

View → **Master Slide** opens Impress's own master-editing mode, showing a master's title and content placeholders, background, and overall theme directly. A presentation can hold more than one Master Slide at once — editing a specific Master Slide's title colour, for instance, updates every slide currently using that particular master, the same propagation behaviour as PowerPoint's own Slide Master.

Layouts vs. Master Slides: A Slightly Different Split Than PowerPoint's

PowerPoint nests Layouts directly underneath one specific Slide Master, in a strict parent-child relationship. Impress organizes this a little differently: its **Layouts** panel (in the sidebar) offers a smaller, simpler set of placeholder arrangements — Title Slide, Title Content, Title Only, Blank — applied to a slide somewhat independently of which Master Slide that slide happens to be using, rather than as a Layout strictly belonging to one specific Master the way PowerPoint structures it. The overall goal is the same; the organizational relationship between "overall theme" and "placeholder arrangement" simply isn't a perfect one-to-one match between the two applications.

Applying a Master Slide to Specific Slides

In the Master Slide panel, right-clicking a specific master offers **Apply to Selected Slides** or **Apply to All Slides** — letting different slides within the same presentation use genuinely different Master Slides, the same underlying capability PowerPoint offers when a single presentation file contains more than one Slide Master.

Placeholders Still Work the Same Way

The core placeholder concept from Word & PowerPoint Chapter 8 — a placeholder's position and default formatting come from the shared master/layout, while its actual content is filled in per-slide — is completely unchanged in Impress. This part of the mental model travels across with zero translation needed.

	POWERPOINT	IMPRESS
Overall theme (background, colours, fonts)	Slide Master	Master Slide
Placeholder arrangement	A Layout, nested under one specific Master	A Layout, applied somewhat independently of the current Master Slide
Multiple designs in one file	Multiple Slide Masters, each with its own Layouts	Multiple Master Slides, applied to selected slides
Direct formatting overriding the shared definition	Disconnects that slide from future Master edits	Disconnects that slide from future Master Slide edits — identical risk

THE DIRECT-FORMATTING-OVERRIDE RISK FROM WORD & POWERPOINT CHAPTER 8 IS UNCHANGED

Manually overriding a placeholder's formatting directly on one specific slide breaks that slide's connection to future Master Slide edits for that one property, in Impress exactly as in PowerPoint — this isn't a new Impress-specific behaviour, it's the same underlying mechanism Word & PowerPoint Chapter 8 already warned about, just under a slightly different set of panel names.

MASTER SLIDES AND LAYOUTS DON'T AUTOMATICALLY COORDINATE THE WAY POWERPOINT'S NESTED STRUCTURE DOES

Because Impress's Layouts panel operates somewhat independently of whichever Master Slide is currently applied, switching a slide's Master Slide does not automatically pick a sensible matching Layout for it, and vice versa — changing one doesn't reliably prompt a reconsideration of the other, the way PowerPoint's strict Master-owns-its-Layouts structure tends to keep the two more visibly connected. After changing either one, it's worth checking that the other still looks right, rather than assuming they'll stay coordinated automatically.

Hands-On Exercises

EXERCISE 1

Open View > Master Slide, and change the title placeholder's colour on the current Master Slide. Return to Normal view and confirm which slides updated. Explain why this matches Word & PowerPoint Chapter 8's own Slide Master exercise almost exactly.

View solution

EXERCISE 2

On one specific slide, manually recolour the title text directly (not through Master Slide view). Then change the Master Slide's title colour again to something different. Describe what happens to that one slide versus every other slide, and explain why.

View solution

EXERCISE 3

A colleague coming from PowerPoint expects that applying a new Master Slide to a slide will automatically also update its placeholder arrangement to something sensible for that master, the way switching a PowerPoint Slide Master's own nested Layouts tends to. Explain why this expectation doesn't necessarily hold in Impress, and what they should check after applying a new Master Slide.

View solution

CHAPTER 7 QUICK REFERENCE

- **View → Master Slide** — Impress's equivalent of PowerPoint's Slide Master, editing the shared theme
- **Layouts panel** — placeholder arrangements, applied somewhat independently of the current Master Slide, unlike PowerPoint's strict Master-owns-Layouts nesting
- A presentation can use **multiple Master Slides**, applied to selected slides via right-click
- Placeholders work identically — position/formatting from the shared definition, content filled in per-slide
- Direct formatting overriding a placeholder disconnects that slide from future Master Slide edits — the same risk as PowerPoint
- Changing a Master Slide or a Layout doesn't automatically coordinate the other — check both after either change

Word & PowerPoint Chapter 11 answered a question this whole course has circled back to repeatedly: where does a presenter's own extra detail belong, if not on the visible slide? Impress answers it the same way, under its own name — the **Presenter Console**. This chapter also flags one genuine, honest gap between the two applications worth knowing about before you need it.

The Same Concept: What the Presenter Sees vs. What the Audience Sees

On a dual-monitor setup, the audience's screen shows only the current slide, full-screen — exactly as designed. The presenter's own screen shows considerably more, privately, exactly the split Word & PowerPoint Chapter 11 established for PowerPoint's own Presenter View.

The Presenter Console

With a second display detected, Impress's **Presenter Console** shows the presenter, on their own screen: the current slide, a preview of the next slide, Speaker Notes, and an elapsed-time clock — the same four pieces of information PowerPoint's Presenter View provides, just under Impress's own name for the same feature.

Speaker Notes in Impress

View → **Notes** attaches private text to an individual slide — the same concept as Word & PowerPoint Chapter 11's Speaker Notes, visible only through the Presenter Console or when printed separately as reference notes pages, never to the live audience.

Rehearse Timings

Slide Show → **Rehearse Timings** runs through the deck in presentation mode while recording how long you actually spend on each slide — the same practice-and-record behaviour Word & PowerPoint Chapter 11 covered for setting up realistic, timed automatic advancing based on a genuine rehearsal rather than a guessed duration.

Setting Up the Slide Show

Slide Show → Slide Show Settings offers a "Loop and repeat after" option for continuous, unattended looping — functionally the same job as PowerPoint's own "Browsed at a kiosk" mode — organized here as a checkbox within one settings dialog rather than PowerPoint's separate set of named presentation-mode options.

Exporting

File → Export As → Export as PDF produces a static, printable reference — the same job as PowerPoint's own Create Handouts, and just as robust and reliable in Impress. For a self-contained *video* export, though, it's worth being upfront: Impress's own video-export support has historically been less mature and less consistently available across versions than PowerPoint's dedicated Create a Video feature. If a genuine video file is specifically needed, it's worth verifying your exact Impress version's current export options directly rather than assuming full feature parity with PowerPoint here.

FEATURE	POWERPOINT	IMPRESS
Private presenter view	Presenter View	Presenter Console — same four elements
Private per-slide notes	Speaker Notes	Notes (View → Notes) — same concept
Recording real per-slide timing	Rehearse Timings	Rehearse Timings — same name, same job
Unattended, looping playback	"Browsed at a kiosk" mode	"Loop and repeat after" in Slide Show Settings
PDF export/handouts	Create Handouts	Export as PDF — equally robust
Self-contained video export	A dedicated, mature Create a Video feature	Less consistent — verify your specific version's support before relying on it

THE SAME ANSWER TO THE SAME QUESTION THIS COURSE HAS ASKED TWICE ALREADY

Word & PowerPoint Chapter 9 argued that detailed content shouldn't clutter a visible slide; Chapter 11 delivered the answer — Speaker Notes plus Presenter View. Impress gives the identical answer, through Notes plus the Presenter Console. The underlying discipline (audience sees a clean slide, presenter has full private detail close at hand) is the same lesson in both applications, not something to relearn.

TEST THE PRESENTER CONSOLE ON THE ACTUAL EQUIPMENT BEFOREHAND — SAME RISK AS POWERPOINT

A dual-monitor setup that isn't configured correctly can expose the presenter's own notes, next-slide preview, and timer to the audience instead of a clean full-screen slide — the exact same risk Word & PowerPoint Chapter 11 warned about for PowerPoint's Presenter View, not a new Impress-specific concern. Test on the real projector or monitor before presenting live, every time. Separately, if a video export is genuinely required for this presentation, confirm your Impress version actually supports it well before the day you need it — this is not something to discover for the first time under time pressure.

Hands-On Exercises

EXERCISE 1

Add Speaker Notes to a slide explaining detail that shouldn't be visible during the live talk. Open the Presenter Console (on a dual-monitor setup, or simulate it) and confirm exactly what the presenter sees versus what the audience-facing screen shows. Identify which Word & PowerPoint chapter's exercise this most directly mirrors.

View solution

EXERCISE 2

You need a presentation to loop continuously and unattended at a trade show booth. Describe exactly which setting in Slide Show Settings accomplishes this, and how this compares structurally to how PowerPoint organizes the same capability.

View solution

EXERCISE 3

A colleague wants to send a self-contained video of their Impress presentation to stakeholders who couldn't attend live, assuming Impress can do this exactly like PowerPoint's Create a Video feature. Explain what they should actually verify first, and what alternative export this chapter confirms is reliably available in the meantime.

View solution

CHAPTER 8 QUICK REFERENCE

- **Presenter Console** — current slide, next-slide preview, Notes, and a timer for the presenter; the audience sees only the clean current slide
- **View → Notes** — private per-slide detail, the same concept as PowerPoint's Speaker Notes
- **Rehearse Timings** — same name, same job as PowerPoint's own feature
- **Slide Show Settings → "Loop and repeat after"** — Impress's equivalent of "Browsed at a kiosk"
- **Export as PDF** — robust and reliable, matching PowerPoint's Handouts export
- Video export support is less consistent than PowerPoint's — verify your specific version before relying on it
- Always test the Presenter Console on the actual presentation equipment beforehand

Chapter 1 raised file format interoperability as an abstract concern. Chapters 2, 5, and 8 each flagged one specific concrete instance of it in passing — a style-name mismatch, an argument-separator translation, a video-export gap. This chapter is where all of that gets addressed directly and systematically: what genuinely survives a round-trip between ODF and Microsoft's own formats cleanly, and what commonly doesn't.

Why This Topic Gets Its Own Chapter

Most of the world still works in Microsoft's own file formats. ODF and OOXML are genuinely different formats, built by different organisations, attempting to represent overlapping but not identical feature sets — every round-trip between them carries some real, if usually small, risk of an imperfect translation. This isn't a flaw specific to LibreOffice; it's an unavoidable consequence of two independently-designed formats trying to represent each other's features.

What Typically Survives Cleanly

For the vast majority of everyday content, opening and saving back and forth between formats causes no visible problems at all: plain text, standard paragraph and character formatting, most Styles, most formulas and functions, ordinary charts, standard slide layouts, and images all translate reliably in both directions. The genuine risk areas are narrower and more specific than "everything might break" — they cluster around a handful of more advanced features.

Word Documents: What Commonly Shifts

- Some advanced, Word-specific field code types without an exact ODF equivalent can convert to static text or a simplified field, losing their "live" behaviour.
- Very elaborate multi-level list numbering schemes occasionally don't translate perfectly.
- Some advanced diagram types can convert to a flattened, non-editable image rather than staying as an editable object.

Spreadsheets: What Commonly Shifts

- A genuinely new or unusual function that one application's version hasn't caught up to yet (Chapter 5's own point) — the function itself, not just its argument separator.
- Certain complex conditional formatting rules or specific chart subtypes may not map one-to-one, shifting slightly in appearance.

Presentations: What Commonly Shifts

- A custom animation effect or timing without an exact equivalent typically simplifies to the closest available effect, rather than an identical one.
- Certain transition types may not have an exact match.
- Embedded video or audio may not play back identically depending on what codecs are actually installed on the machine opening the file.

A Genuinely Big One: Macros Are Not Portable

This deserves its own section, separate from the smaller shifts above, because it's a fundamentally different kind of problem. Excel Advanced Chapters 6 through 8 covered VBA (Visual Basic for Applications) — Microsoft Office's own macro language. LibreOffice's own macro language, **LibreOffice Basic**, is a genuinely different language. A `.xlsm` workbook's VBA macros generally do **not** run natively in Calc — this isn't a formatting quirk that mostly survives with minor shifts, it's a completely different programming language that requires deliberate rewriting, not just opening the file and expecting it to work.

Best Practices for Working Across Both Formats

- Keep a genuinely final or archival copy in whichever format it actually needs to stay compatible with long-term.
- After any round-trip, spot-check the specific elements most likely to shift — complex fields, unusual chart types, and especially any macros — rather than assuming a clean open means everything survived.
- When collaborating with people using a different suite, agree explicitly up front on which format is the genuine "source of truth" for that document, rather than letting the format drift unintentionally with each person's own save habits.

APP TYPE	USUALLY SURVIVES CLEANLY	COMMONLY SHIFTS
Writer / Word	Text, standard formatting, most Styles	Advanced field codes, elaborate list numbering, some diagrams
Calc / Excel	Formulas, common functions, standard charts	Very new/unusual functions, some conditional formatting, some chart subtypes
Impress / PowerPoint	Standard layouts, common transitions, images	Custom animation timings, some transition types, embedded media codecs
All three (macros)	Nothing — genuinely different languages	VBA and LibreOffice Basic require manual rewriting, not just a re-open

THIS CHAPTER IS WHY SEVERAL EARLIER CHAPTERS' WARNINGS EXISTED

Chapter 2's style-name-mismatch warning, Chapter 5's separator-translation note, and Chapter 8's video-export caution weren't isolated quirks — each was one specific instance of exactly this chapter's general principle. Recognising that pattern (two genuinely different formats, overlapping but not identical feature sets, translated as faithfully as possible but not always perfectly) is worth more than memorising every individual example.

IF A WORKBOOK RELIES ON VBA MACROS, OPENING IT IN CALC WILL NOT "JUST WORK"

This is the single most important practical takeaway of this entire chapter: unlike a font shift or a chart-type change, VBA macros from Excel Advanced Chapters 6-8 do not run in LibreOffice without being deliberately rewritten in LibreOffice Basic. If a document, workbook, or presentation you're moving to LibreOffice depends on macros, budget real time for rewriting them — this is not a "check and confirm it's fine" situation the way most of this chapter's other examples are, it's a genuine rewrite.

Hands-On Exercises

EXERCISE 1

A .xlsx workbook contains only formulas, standard formatting, and a basic column chart — no macros, no unusual functions. Based on this chapter, predict whether opening and re-saving it in Calc is likely to cause any visible changes, and explain your reasoning.

View solution

EXERCISE 2

A colleague wants to move a .xlsm workbook containing several VBA macros (built following Excel Advanced Chapters 6-8) entirely over to LibreOffice Calc. Explain exactly what will and won't work immediately after opening it, and what they'd actually need to do to get the macros working again.

View solution

EXERCISE 3

Your team is genuinely split between LibreOffice and Microsoft Office users collaborating on the same set of documents. Describe the best-practice habit this chapter recommends for avoiding format-drift confusion, and explain why "just let everyone save in whatever they're using" is a risky default.

View solution

CHAPTER 9 QUICK REFERENCE

- Most everyday content — text, standard formatting, common formulas, standard charts — survives a round-trip cleanly
- Advanced field codes, elaborate list numbering, and some diagrams are the main Word/Writer risk areas
- Very new/unusual functions and some conditional formatting/chart subtypes are the main Calc/Excel risk areas
- Custom animation timings, transitions, and embedded media codecs are the main Impress/PowerPoint risk areas
- **VBA macros do not run in LibreOffice** — LibreOffice Basic is a genuinely different language, requiring a real rewrite, not just a re-open
- Always spot-check complex fields, unusual charts, and any macros after a round-trip — don't assume a clean-looking open means everything survived

Word & PowerPoint Chapter 12 built a styled, mail-merged business report and its companion presentation deck. This capstone rebuilds the exact same project, entirely in Writer, Calc, and Impress — and at every step, names explicitly whether that piece transferred directly, needed a genuinely different approach, or leans on general suite knowledge this course didn't cover in its own dedicated chapter.

The Project Brief

The same brief as before: a professionally structured report, reviewed by a colleague, personalized to several named recipients, and summarized into a short, disciplined slide deck for a live meeting — this time built start to finish without Word or PowerPoint at all.

Step 1 — Structuring the Report with Writer's Styles

Every section title uses a real paragraph style — Heading 1 for major sections, applied via the Styles panel (F11) — never manually bolded text. The Navigator (F5) confirms the structure is genuinely present, not just visually convincing. *(Chapter 2 — transfers directly, different panel.)*

Step 2 — Page Layout with Page Styles Instead of Section Breaks

This is the one genuinely different mechanism in the whole rebuild. The cover page uses the built-in "First Page" style; a wide data table gets its own isolated page using the "Landscape" style, entered and exited with two separate page breaks (Insert → More Breaks → Insert Page Break) rather than Word's own paired section breaks. Because a Page Style already owns its own header definition, there's no separate "Link to Previous" step needed at all — switching Page Styles handles both the orientation change and the header difference in one unified mechanism. *(Chapter 3 — the standout structural difference in this entire course.)*

Step 3 — Table of Contents, Captions & Cross-References

Writer's own **Insert → Table of Contents and Index** and **Insert → Caption** commands solve the identical job Word & PowerPoint Chapter 4 covered — a live ToC built from Heading styles, auto-numbered figures and tables. This course didn't dedicate its own chapter to this specific topic, but the underlying commands work on the same principle: real structure in, a live, updatable reference out. *(General suite knowledge, not a dedicated chapter here.)*

Step 4 — Collaborative Review with Track Changes and Comments

Edit → **Track Changes** → **Record Changes**, plus **Insert** → **Comment**, give a colleague the same reviewing tools Word & PowerPoint Chapter 6 covered — tracked insertions/deletions and margin discussion, reviewed individually rather than blindly accepted. Also not its own dedicated chapter in this course, but functioning essentially identically to Word's own tools. *(General suite knowledge, not a dedicated chapter here.)*

Step 5 — Personalizing via the Mail Merge Wizard

A cover letter is personalized using **Tools → Mail Merge Wizard**, connected to a registered Calc spreadsheet as its data source, with field names matched explicitly against the spreadsheet's own column headers before the full merge runs. *(Chapter 4 — same job, a materially different guided-wizard flow.)*

Step 6 — Building the Companion Deck with the Master Slide

The Impress deck's Master Slide is set to match the Writer report's own accent colour, the same shared-design idea Chapter 1 introduced at the very start of this course. Every slide uses one of a small set of Layouts. *(Chapter 7 — same underlying idea, a somewhat differently organized Master/Layout relationship.)*

Step 7 — Distilling the Report into Slides

The deck does not copy the report's own paragraphs. Each slide covers exactly one takeaway, kept within the 6×6 guideline; the report's own dense data table becomes a single chart. This step is pure design judgment from Word & PowerPoint Chapter 9 — it applies completely unchanged, since it was never actually about which application you're using in the first place. *(Transfers with zero change — a judgment principle, not a tool-specific feature.)*

Step 8 — Purposeful Motion in Impress

Impress's own Animation and Transition panels offer the same underlying categories Word & PowerPoint Chapter 10 covered for PowerPoint — one consistent transition style across the deck, and a single On Click entrance animation for the one slide that genuinely benefits from a progressive reveal. Not its own dedicated chapter in this course, but the same restraint judgment applies regardless of which application implements it. *(General suite knowledge, not a dedicated chapter here — same design judgment as Step 7.)*

Step 9 — Rehearsing and Presenting

Every slide's extra explanatory detail lives in its own Notes, not on the visible slide. The Presenter Console keeps that detail private during the live meeting; Rehearse Timings confirms realistic pacing beforehand. Afterward, the deck is exported confidently as PDF handouts — and, per Chapter 9's own honest caveat, video export is verified against the actual installed Impress version before being relied on for stakeholders who couldn't attend. *(Chapter 8, plus Chapter 9's own interoperability caveat.)*

Putting It All Together

CAPSTONE PIECE	TECHNIQUE USED	COVERAGE
Report section structure	Writer paragraph Styles, Navigator	Chapter 2 — full chapter
Cover, isolated landscape page, headers	Page Styles, not section breaks	Chapter 3 — full chapter, the key structural difference
Navigable ToC, numbered figures/tables	Insert Table of Contents, Insert Caption	General suite knowledge
Colleague review before finalizing	Track Changes, Comments	General suite knowledge
Personalized cover letters	Mail Merge Wizard, a registered Calc source	Chapter 4 — full chapter
A visually consistent deck	Master Slide, Layouts	Chapter 7 — full chapter
One takeaway per slide, a chart over a table	Content density, visual hierarchy	Pure design judgment — unchanged from Word & PowerPoint Ch.9
One purposeful animated reveal	Impress's own Animation/Transition panels	General suite knowledge
Extra detail kept off the visible slide	Notes, Presenter Console, verified PDF/video export	Chapter 8, plus Chapter 9's export caveat

THREE BUCKETS, NOT ONE UNDIFFERENTIATED PILE OF DIFFERENCES

Looking back across this whole course, nearly every concept fell into one of three buckets: (1) the identical idea, reached through a different panel — the overwhelming majority of this course; (2) one genuinely different mental model — Page Styles, Chapter 3; and (3) one genuinely bigger gap worth real caution — macros, Chapter 9. Recognising which bucket a new, unfamiliar LibreOffice feature falls into is a more useful skill going forward than memorising every menu location individually.

IF THIS CAPSTONE HAD NEEDED VBA AUTOMATION, IT WOULD NOT HAVE TRANSFERRED AT ALL

Nothing in this rebuild depended on macros — but if the original Word & PowerPoint project had included, say, a VBA-driven Excel dashboard refresh button (the kind of automation Excel Advanced Chapters 6-8 covered), that specific piece would not have worked simply by opening the workbook in Calc. It would need to be genuinely rewritten in LibreOffice Basic, per Chapter 9's own central warning — the single most consequential exception to this whole course's otherwise reassuring "it mostly just transfers" theme.

Hands-On Exercises

EXERCISE 1

Build the Writer report foundation from Steps 1-2: at least 2 Heading-1 sections using real paragraph styles, and one isolated landscape page (via two page breaks switching Page Style) holding a wide table. Confirm the Navigator shows your sections, and that only the landscape page is affected by the orientation change.

View solution

EXERCISE 2

Using a small Calc spreadsheet (3-4 rows, Name and Amount columns), run the Mail Merge Wizard against it to produce personalized letters. Then build a 3-slide Impress deck using one Master Slide, distilling — not copying — the report's own content into one idea per slide.

View solution

EXERCISE 3

A colleague asks why this capstone's own comparison table marks some steps "Chapter X — full chapter" and others "General suite knowledge, not a dedicated chapter here." Explain the actual difference between these two labels, and why both kinds of step were still worth including in a capstone that's meant to mirror the original Word & PowerPoint project completely.

View solution

COURSE RECAP — LIBREOFFICE

- **Ch.1:** the suite model, ODF vs. proprietary formats, the "same job, different app" framing
- **Ch.2:** Writer's Styles and Navigator — identical to Word & PowerPoint Ch.2, a different panel
- **Ch.3:** Page Styles instead of section breaks — this course's one genuine mental-model difference
- **Ch.4:** the Mail Merge Wizard and registered data sources
- **Ch.5:** Calc's formulas — nearly identical to Excel, with real locale-dependent syntax gotchas
- **Ch.6:** Calc's Pivot Table — the same job, a static Layout dialog instead of a live panel
- **Ch.7:** the Master Slide — Impress's own Slide Master, organized slightly differently
- **Ch.8:** the Presenter Console, Notes, Rehearse Timings, and an honest note on video export
- **Ch.9:** file format interoperability in practice — what survives, what shifts, and why macros are different
- **Ch.10:** all of the above, rebuilding the Word & PowerPoint capstone entirely in LibreOffice