EDUSHARK TRAINING

Career-Focused Programs for Aspiring Analysts

Applied Data Analytics & Business Intelligence

A 12-Week Workbook for Beginners | *Next Edition*

From data foundations to deployed R Shiny dashboards.

Pawan Rama Mali

Instructor

edusharktraining.com +91 8105 683 683

Applied Data Analytics & Business Intelligence: A 12-Week Workbook for Beginners
Next Edition

Instructor & Author: Pawan Rama Mali
Edition: Second (Next), 2026
Publisher: EDUSHARK TRAINING
Web: https://edusharktraining.com
Contact: info@edusharktraining.com | +91 8105 683 683

This is the revised edition of the EDUSHARK Applied Data Analytics workbook. The first edition has been retained alongside this one. This Next Edition adds instructor commentary, beginner-mistakes notes, common interview questions, visual workflow diagrams, and a new appendix on the modern analytics stack.

All trademarks belong to their respective owners. Code samples are released for personal learning use; please credit the source if you republish them.

Typeset with LaTeX using Latin Modern, Helvetica and tcolorbox.

A Letter from Your Instructor

Hello, and welcome.

I am Pawan Rama Mali, and I will be your guide for the next twelve weeks. Before we begin, a few honest words.

This workbook will *not* make you a data analyst on its own. Books rarely do. What it will do is give you a structured, professional path to follow – one that I have refined over multiple cohorts of EDUSHARK learners, some of whom came in without any programming background and walked out with deployed dashboards and offers in hand.

A few things you should know up front:

- **It is okay to be slow.** A few of the early chapters will feel intuitive; statistics and SQL window functions usually do not. Pace yourself. Re-reading a section three times is normal.
- **Type out every code sample by hand.** Copy-pasting feels productive but is the single biggest reason learners forget what they "learned" a week later. There is something about your fingers carrying the syntax that the eyes alone cannot do.
- **Build the projects.** Each week ends with a small portfolio piece. Skipping them is the most common reason learners stall around Week 6.
- **Ask for help early.** If you are stuck for more than 30 minutes on the same error, post a question – in the cohort forum, on Stack Overflow, or to a friend. Spinning wheels is not learning.
- **Read your code aloud before you ship it.** If you cannot narrate what each line does to a colleague, you do not yet understand it – and an interviewer will catch the gap in the first five minutes.

Lastly: the goal of this workbook is not to make you sound like an analyst. It is to make you *able to do the work of one*. The two are not the same. Companies pay for the second.

See you in Chapter 1.

— Pawan

How to Use This Workbook

Each chapter covers one week (seven days) of study. The structure is the same throughout, which is deliberate – you should not need to figure out the layout every time. Here is what to expect on every chapter.

The recurring elements

- **Prerequisites & study time strip** (small grey box, top of the chapter): what you should already know and roughly how long the week will take.
- **Learning Objectives** (deep blue box): what you will be able to do by the end of the chapter.
- **From the Instructor** (purple box, new): short, personal notes from me about what tripped up earlier cohorts and what made the difference for them.
- **Concept boxes** (light blue): a definition or core idea worth memorising.
- **Pro Tip** (mint): a small piece of practitioner wisdom.
- **Watch Out** (coral red): a common pitfall.
- **Try It Yourself** (gold): a short exercise. Try it before peeking at the solution at the back of the workbook.
- **Mistakes Beginners Make** (grey, new): an end-of-chapter list of the most common errors and how to recognise them when they happen to you.
- **Interview Questions You'll Be Asked** (sky blue, new): real questions taken from analyst interviews – start practising them now.
- **End-of-Week Project** (cream card): a small portfolio piece that ties the week together.
- **Key Takeaways** (mint card): one-line summaries for quick review before exams or interviews.

Tools you will install

You will install free tools one at a time, when each is introduced: Microsoft Excel or Google Sheets, MySQL or PostgreSQL, R, RStudio Desktop, optionally Power BI Desktop and Tableau Public. None require a paid licence to follow this book. The first time a SQL dialect-specific feature appears, you will see a small MySQL or PostgreSQL tag next to it – the book is careful never to mix the two without telling you.

Companion materials

The course website hosts:

- Per-week blog posts mirroring each chapter.
- Sample datasets in `/data/` for every week.

- A separate *Exercises & Solutions* workbook with hundreds of practice problems.

What's new in this Next Edition

If you read the first edition, the structure is unchanged but you will notice:

- Instructor commentary in every chapter.
- A "Mistakes Beginners Make" box at the end of each chapter.
- "Common Interview Questions" added per chapter.
- Visual workflow diagrams for analytics lifecycle, SQL joins, star schema, Shiny reactivity, and A/B test flow.
- Explicit SQL dialect tags wherever it matters.
- A new appendix on the modern analytics stack (DuckDB, Polars, dbt, semantic layers, cloud warehouses).
- Softer, more honest framing around career outcomes – this is a *career-focused* workbook, not a magic pill.

Now turn the page.

Contents

A Letter from Your Instructor **i**

How to Use This Workbook **iii**

1 Data Foundations & Excel Analytics **1**

1.1 Day 1 – Introduction to Data Analytics . . . 1
1.2 Day 2 – Data Types and Structures . . . 2
1.3 Day 3 – Business Questions & Problem Framing . . . 3
1.4 Day 4 – KPIs & Metrics Design . . . 4
1.5 Day 5 – Excel Fundamentals for Analytics . . . 4
1.6 Day 6 – Advanced Excel & Pivot Tables . . . 5
1.7 Day 7 – Project: Sales Performance Dashboard . . . 6

2 SQL Fundamentals & Database Basics **9**

2.1 Day 1 – Why Databases Exist . . . 9
2.2 Day 2 – Your First SELECT . . . 10
2.3 Day 3 – Filtering with WHERE . . . 11
2.4 Day 4 – ORDER BY, GROUP BY, HAVING . . . 11
2.5 Day 5 – Functions That Clean Real Data . . . 12
2.6 Day 6 – JOINs . . . 12
2.7 Day 7 – Project: E-commerce SQL Report . . . 13

3 Advanced SQL & Analytics Queries **15**

3.1 Day 1 – Subqueries . . . 15
3.2 Day 2 – Correlated Subqueries . . . 16
3.3 Day 3 – Window Functions, Part 1: Ranking . . . 16
3.4 Day 4 – Window Functions, Part 2: Running & Comparing . . . 17
3.5 Day 5 – Common Table Expressions (CTEs) . . . 18
3.6 Day 6 – Query Optimisation, the Beginner's Edition . . . 18
3.7 Day 7 – Project: Advanced Analytics Report . . . 19

4 Statistics Foundations for Data Analysis **21**

4.1 Day 1 – Central Tendency . . . 21
4.2 Day 2 – Dispersion . . . 22
4.3 Day 3 – Shape: Distributions . . . 23
4.4 Day 4 – Probability Fundamentals . . . 23
4.5 Day 5 – Common Distributions . . . 24
4.6 Day 6 – Sampling & Estimation . . . 24
4.7 Day 7 – Project: Statistical Analysis Report . . . 25

5 Advanced Statistics & EDA **27**
5.1 Day 1 – Hypothesis Testing . . . 27
5.2 Day 2 – The Common Tests . . . 28
5.3 Day 3 – Correlation, not Causation . . . 28
5.4 Day 4 – Exploratory Data Analysis . . . 29
5.5 Day 5 – A/B Testing . . . 29
5.6 Day 6 – Visualising for Analysis . . . 30
5.7 Day 7 – Project: EDA & Statistical Analysis . . . 31

6 R Programming Fundamentals **33**
6.1 Day 1 – R, RStudio and Hello World . . . 33
6.2 Day 2 – Data Types and Structures . . . 34
6.3 Day 3 – Control Flow & Functions . . . 35
6.4 Day 4 – The Tidyverse and the Pipe . . . 35
6.5 Day 5 – `dplyr`: Five Verbs Run Half Your Job . . . 36
6.6 Day 6 – Reshaping with `tidyr` & Joins . . . 36
6.7 Day 7 – Project: Data Wrangling Pipeline . . . 37

7 Advanced R & Data Visualisation **39**
7.1 Day 1 – ggplot2 Fundamentals . . . 39
7.2 Day 2 – Faceting, Themes, Palettes . . . 40
7.3 Day 3 – Strings with `stringr` . . . 41
7.4 Day 4 – Dates with `lubridate` . . . 41
7.5 Day 5 – Advanced dplyr . . . 42
7.6 Day 6 – Project Hygiene . . . 42
7.7 Day 7 – Project: R Analytics Report . . . 43

8 R Shiny – Interactive Dashboards **45**
8.1 Day 1 – Hello, Shiny . . . 45
8.2 Day 2 – UI Components and Layout . . . 46
8.3 Day 3 – Reactivity Deep Dive . . . 47
8.4 Day 4 – Interactive Visualisations . . . 48
8.5 Day 5 – shinydashboard & bslib . . . 48
8.6 Day 6 – Deployment . . . 49
8.7 Day 7 – Project: Interactive Business Dashboard . . . 50

9 Power BI Fundamentals (Optional Track) **53**
9.1 Day 1 – The Power BI Stack . . . 53
9.2 Day 2 – Connecting to Data . . . 54
9.3 Day 3 – Power Query . . . 54
9.4 Day 4 – Data Modelling . . . 55
9.5 Day 5 – Visualisations & Reports . . . 56
9.6 Day 6 – DAX Fundamentals . . . 56
9.7 Day 7 – Project: Power BI Sales Dashboard . . . 58

10 Tableau Essentials (Optional Track) 61
10.1 Day 1 – Tableau, in 10 Minutes . . . 61
10.2 Day 2 – Basic Visualisations . . . 62
10.3 Day 3 – Advanced Visuals . . . 62
10.4 Day 4 – Calculated Fields & Parameters . . . 63
10.5 Day 5 – Dashboards . . . 64
10.6 Day 6 – Publishing & Comparing Tools . . . 64
10.7 Day 7 – Project: Published Tableau Dashboard . . . 65

11 Capstone Project – Part 1 67
11.1 Day 1 – Choose Your Project . . . 67
11.2 Day 2 – Sourcing Data & Setting Up Git . . . 68
11.3 Day 3 – Data Preparation & SQL Pipeline . . . 69
11.4 Day 4 – Statistical Analysis in R . . . 70
11.5 Day 5 – Shiny App, Part 1 – Skeleton . . . 71
11.6 Day 6 – Shiny App, Part 2 – Polish . . . 71
11.7 Day 7 – Checkpoint Review . . . 72

12 Capstone Completion & Career Prep 75
12.1 Day 1 – Ship It . . . 75
12.2 Day 2 – Documentation & Portfolio . . . 76
12.3 Day 3 – Resume & LinkedIn . . . 77
12.4 Day 4 – Technical Interviews . . . 78
12.5 Day 5 – Behavioural Interviews . . . 78
12.6 Day 6 – Presentation Practice . . . 79
12.7 Day 7 – Graduation . . . 80

A Glossary 83

B Further Resources 87
B.1 Free books worth your time . . . 87
B.2 Online practice . . . 87
B.3 Cheat-sheets and references . . . 87
B.4 Communities . . . 87
B.5 Datasets to keep practising on . . . 88
B.6 Newsletters & podcasts . . . 88
B.7 Where to host your portfolio . . . 88

C The Modern Analytics Landscape 89
C.1 The shape of a modern analytics stack . . . 89
C.2 Faster local analytics: DuckDB, Polars, Arrow, Parquet . . . 89
C.3 Transformation: dbt and friends . . . 90
C.4 Cloud warehouses . . . 90
C.5 Semantic layers . . . 91
C.6 Python for analysts (a short note) . . . 91
C.7 Workflow / orchestration: things that schedule everything . . . 91
C.8 What to learn next, and in what order . . . 91

CHAPTER 1

Data Foundations & Excel Analytics

Week 1

Prerequisites: None. If you can open a spreadsheet, you are ready.
Estimated study time: 6–8 hours total across the week (about an hour a day).
Tooling / dialect note: Excel 365 (or LibreOffice Calc / Google Sheets as a substitute).

OBJECTIVES

By the end of this week you will be able to:

1. Explain what data analytics is and why companies pay for it.
2. Classify any data sample by structure and by measurement type.
3. Convert a vague business request into a SMART analytical question.
4. Design meaningful KPIs for a department or product line.
5. Use Excel formulas, lookups and Pivot Tables to produce a one-page sales analysis dashboard.

From the Instructor – Pawan

The first week is the most underrated of the twelve. Most of my learners arrive thinking they will "learn the tools", skim the Excel section as something they already know, and then stall in Week 4 because they cannot frame a clear question to answer. *Spend real time on Days 3 and 4 – SMART questions and KPI design.* Those two days are the difference between an analyst and a person who runs SQL queries on demand.

1.1 Day 1 – Introduction to Data Analytics

Day 1 What it is, what it pays, and where it lives.

Data analytics is the discipline of turning raw observations into decisions. A retailer logs every till transaction; an analyst studies those logs and answers: *Which stores beat their target? Which products are about to stock out? Which discount actually increased profit?* The technical toolkit is interesting only because the answers are.

Concept

Data analytics is the process of *collecting, cleaning, transforming, visualising and modelling* data so that humans can make better decisions, faster and with less guesswork.

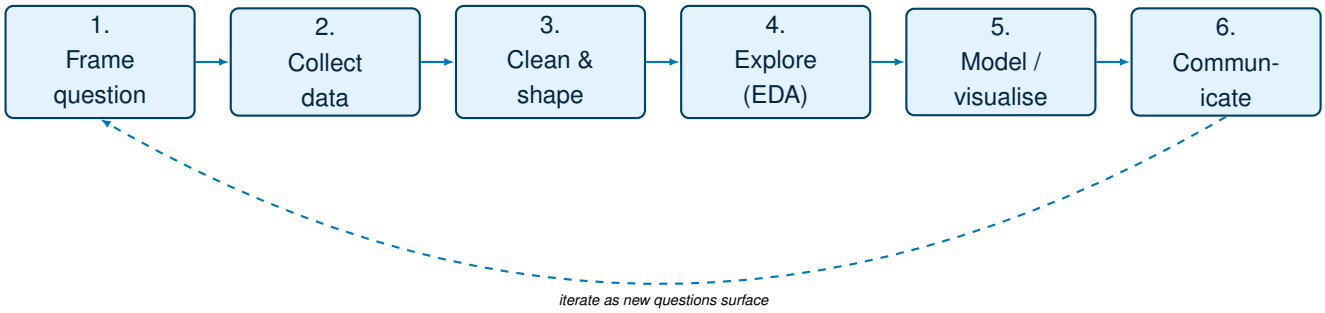
1.1.1 The four flavours of analytics

Every analytics question reduces to one of four shapes. Knowing which shape you are tackling tells you what tools you need.

Type	Question	Example	Tools
Descriptive	What happened?	Sales fell 15% last quarter.	Excel, SQL
Diagnostic	Why did it happen?	Top SKU went out of stock for 3 weeks.	SQL, EDA
Predictive	What will happen?	Next-quarter forecast: -8% if no action.	R, Python
Prescriptive	What should we do?	Re-order SKU-12 every 14 days.	R, OR

1.1.2 The analytics lifecycle

A real project moves through six stages. Beginners often jump straight to step 5; experienced analysts spend most of their time in steps 1 to 3, because those decide whether the rest will be useful.



Pro Tip

If you can't explain the question in one sentence to a non-technical stakeholder, do not start work yet. Define the question first.

1.1.3 Where analysts work

You will find analysts in five large industries that hire most freshers: **BFSI** (banks, insurance), **Healthcare & Pharma**, **Retail & E-commerce**, **IT & Product**, and **Operations / Supply Chain**. Each has its own vocabulary – "churn" for telecom, "shrinkage" for retail, "adverse event" for pharma – but the analytical method is the same.

1.2 Day 2 – Data Types and Structures

Day 2 If you can't classify your data, you can't pick the right tool.

1.2.1 Structure: how is it stored?

- **Structured** – rows and columns with a fixed schema. Excel, SQL tables, CSVs.
- **Semi-structured** – has hierarchy but no fixed schema. JSON, XML, log files.
- **Unstructured** – free-form text, audio, images, PDF reports.

1.2.2 Measurement: what kind of value is it?

Family	Sub-type and definition	Example
Numerical	Discrete (countable integers)	# of orders
Numerical	Continuous (any real value)	temperature, weight
Categorical	Nominal (no order)	city, gender
Categorical	Ordinal (ranked)	rating 1–5, T-shirt size
Date/time	Point or interval in time	2026-05-01 14:23

Watch Out

Storing an ordinal as a plain text label ("Low", "Medium", "High") means your tools will sort it alphabetically. Always encode order explicitly – either with numeric codes or with R/Pandas *ordered factors*.

1.2.3 The six dimensions of data quality

Accuracy, Completeness, Consistency, Timeliness, Uniqueness, Validity. Memorise the list – almost every messy dataset is broken on at least two of these dimensions. Your first job on any new file is a quick quality audit.

1.3 Day 3 – Business Questions & Problem Framing

Day 3 Vague requests are the analyst's biggest time-sink.

"Can you pull some numbers on the campaign?" is not a question. It is a starting point. Your value as an analyst is to turn that into something answerable.

Concept

The **SMART** framework: a good analytical question is **S**pecific, **M**easurable, **A**ction-oriented, **R**elevant, and **T**ime-bound.

Vague: "How is the new campaign doing?"

SMART: "By how many percentage points did the May 2026 Mother's Day email increase 14-day repeat-purchase rate among customers who opened the email, versus a matched control cohort, and is the lift statistically significant?"

1.3.1 The five-question stakeholder interview

Before writing a single query, ask:

1. What decision will be made with this analysis?
2. Who is the audience and what do they already believe?
3. By when is it needed – and what changes if it slips?
4. What does "good" look like? What would change your mind?
5. What data do we already have, and where does it live?

1.4 Day 4 – KPIs & Metrics Design

Day 4 The numbers your dashboard will live or die by.

Concept

A **metric** is any quantity you measure. A **KPI** (Key Performance Indicator) is a metric tied to a goal, with a target and an owner. "Page views" is a metric. "Monthly active users $\geq$ 200,000" is a KPI.

1.4.1 Leading vs lagging

Lagging indicators measure outcomes that have already happened: revenue, churn, profit. **Leading** indicators measure inputs that predict the outcome: trial signups, NPS, sales calls booked. A good dashboard pairs each lagging KPI with one or two leading indicators.

1.4.2 A library of starter KPIs

Domain	KPIs you will be expected to know
Sales	Revenue, conversion rate, AOV (avg. order value), CAC, LTV, win rate
Marketing	CTR, CPC, ROAS, engagement rate, MQL→SQL, attribution
Operations	On-time-in-full, utilisation, cycle time, defect rate
Finance	ROI, gross margin, operating margin, cash burn, runway
Product	DAU/MAU, retention curve, activation rate, NPS

Try It Yourself

Pick a business you use weekly (a streaming service, a food-delivery app, your favourite e-commerce site). Write down three lagging and three leading KPIs you would put on the founder's dashboard. Justify each in one line.

1.5 Day 5 – Excel Fundamentals for Analytics

Day 5 The first analytics tool. Underrated, ubiquitous, very fast.

Treat Excel as the warm-up tool: it teaches you cell references, formulas, and structured data, all of which translate directly to SQL and R.

1.5.1 The essentials, by function family

Family	Functions
Aggregate	`SUM, AVERAGE, COUNT, COUNTA, MIN, MAX, MEDIAN, MODE`
Logical	`IF, AND, OR, NOT, IFS, IFERROR`
Text	`LEFT, RIGHT, MID, LEN, TRIM, CONCAT, TEXT`
Date	`TODAY, NOW, DATE, YEAR, MONTH, DAY, EOMONTH, NETWORKDAYS`
Reference	`VLOOKUP, HLOOKUP, INDEX-MATCH, XLOOKUP`

1.5.2 Cell references, the part everyone gets wrong

- `A1` – relative; shifts when you copy.
- `$A$1` – absolute; never shifts.
- `$A1` or `A$1` – mixed; column or row locked.

Watch Out

Forgetting to lock a lookup range with `$` is the single most common cause of "my dashboard mysteriously breaks when I copy a formula down."

1.6 Day 6 – Advanced Excel & Pivot Tables

Day 6 Where Excel earns its keep.

1.6.1 Lookups: the right one

VLOOKUP works but is brittle (column number is positional). **INDEX-MATCH** is the classical replacement: `=INDEX(price_col, MATCH(lookup_value, key_col, 0))`. Since Excel 365, **XLOOKUP** is the modern one-line answer: `=XLOOKUP(lookup, keys, prices, "not found")`.

1.6.2 Conditional aggregates

`SUMIFS`, `COUNTIFS`, and `AVERAGEIFS` take pairs of *(range, criterion)*. They replace 80% of the "please filter and then sum" tasks that people do by hand.

1.6.3 Pivot Tables in 60 seconds

1. Convert your data to a Table (`Ctrl+T`).
2. `Insert` → `PivotTable`.
3. Drag categorical fields into *Rows* or *Columns*; numeric fields into *Values*.
4. Change the aggregation in *Value Field Settings* (sum, average, count, ...).

5. Add Slicers for one-click filtering.

Pro Tip

A Pivot Table is just SQL's `GROUP BY` with a mouse. Once you can build one in your sleep, week 2 will feel familiar.

1.6.4 The five Excel formulas you'll re-use forever

Memorise these patterns. They cover most one-page dashboards.

```
=XLOOKUP(A2, products[id], products[price], "not found")
=SUMIFS(orders[amount], orders[region], $E2, orders[month], F$1)
=AVERAGEIFS(orders[amount], orders[status], "completed")
=COUNTIFS(orders[customer_id], A2, orders[status], "<>cancelled")
=IF(B2 >= AVERAGE($B$2:$B$500) + 1.5*STDEV.S($B$2:$B$500),
    "high", "normal")
```

Listing 1.1: *Five Excel patterns*

1.7 Day 7 – Project: Sales Performance Dashboard

End-of-Week Project

Build a one-page Excel dashboard from `week1-sales-data.csv` (provided) that answers:

1. Which region grew fastest in the last 90 days?
2. Which three products contribute most to total revenue?
3. What is the average order value, and how has it moved month-on-month?
4. Pick five KPIs you defined on Day 4 and display them as "KPI cards" (boxed values with last-period delta).
5. Write three actionable recommendations for the sales head.

Deliverable: a single .xlsx file with a `Data` sheet, a `Pivot` sheet, and a `Dashboard` sheet – the last one printable on one A4 page.

Mistakes Beginners Make

1. **Skipping Day 3.** Learners want to "learn formulas" and ignore problem framing. Six weeks later they cannot turn a stakeholder's request into a query.
2. **Confusing metrics with KPIs.** Every KPI is a metric, but most metrics are not KPIs. Without a target and an owner, it is just a number on a dashboard.
3. **Treating ordinal categories as text.** "Low / Medium / High" sorted alphabetically becomes "High / Low / Medium". Encode order explicitly.

4. **Memorising VLOOKUP syntax** instead of internalising the *lookup pattern*. Then they cannot translate it to SQL joins next week.
5. **Locking the wrong cell.** `$A1` vs `A$1` vs `$A$1` – get the difference wrong once and your whole formula column breaks silently.

Interview Questions You'll Be Asked

- "Walk me through how you would frame an analysis if a stakeholder said 'sales are down – can you look into it?' "
- "What is the difference between a metric and a KPI? Give me an example of each."
- "Explain leading vs lagging indicators with one example from a domain you know."
- "In Excel, when would you use INDEX-MATCH instead of XLOOKUP?"
- "Describe a Pivot Table you have built. What does it correspond to in SQL?"

Key Takeaways

- Analytics has four flavours; every question reduces to one of them.
- Always classify your data by structure and measurement before opening any tool.
- SMART beats vague every time.
- Lagging KPIs measure outcomes; leading KPIs predict them. Pair them.
- Excel is your warm-up: master cell references, conditional aggregates, and Pivot Tables.

Up next: we move from a single spreadsheet to a real database, where the data is too large to scroll through and the queries are where the analysis actually happens.

CHAPTER 2

SQL Fundamentals & Database Basics

Week 2

Prerequisites: Chapter 1 completed – you can describe a SMART question and read a spreadsheet.
Estimated study time: 8–10 hours. SQL is the language you will use the most – do not rush.
Tooling / dialect note: MySQL Workbench or pgAdmin (free). MySQL-specific syntax is tagged MySQL; PostgreSQL-specific syntax is tagged PostgreSQL.

OBJECTIVES

By the end of this week you will:

1. Read an Entity–Relationship diagram and identify primary / foreign keys.
2. Write `SELECT` queries with `WHERE`, `ORDER BY`, `LIMIT`.
3. Group, count, sum and filter aggregates with `GROUP BY` and `HAVING`.
4. Use string, numeric and date functions for cleanup.
5. Combine tables with `INNER`, `LEFT` and `RIGHT JOIN`s.
6. Deliver an e-commerce SQL analysis report.

From the Instructor – Pawan

SQL is the only skill in this book that almost every analyst interview will test live, on a whiteboard or a shared screen. Recruiters trust SQL because it is hard to fake. Treat this week as a marathon, not a sprint. If you finish the project comfortably, do five more LeetCode-Easy SQL problems before moving on – you will thank yourself in Week 12.

2.1 Day 1 – Why Databases Exist

Day 1 Spreadsheets stop being the right tool when data grows.

Modern Excel handles over a million rows, and it is wonderful for one analyst working on a single file. Trouble starts when several people need to read and write the same data at once, when the file gets too large to open quickly, when changes need to be tracked, or when the same question has to be answered every morning. That is the job of a **database**.

A database is structured storage that lets many people read and write concurrently without losing

data, and answers questions in milliseconds once it is indexed properly. The most common kind is a **relational database** (or RDBMS), where data lives in tables of rows and columns linked to each other by **keys**.

Concept

A **primary key** is a column (or set of columns) that uniquely identifies every row in a table. A **foreign key** is a column that references the primary key of another table, creating a relationship.

2.1.1 The five tables every e-commerce store has

- `customers` (one row per person)
- `products` (one row per SKU)
- `orders` (one row per checkout)
- `order_items` (one row per product within an order)
- `categories` (one row per product category)

`orders.customer_id` is a foreign key pointing at `customers.customer_id`. `order_items.order_id` points at `orders.order_id`. Most analytical SQL is just joining these tables back together.

Pro Tip

Install **MySQL Workbench** (or pgAdmin for Postgres). Load the sample database `week2-ecommerce-database.sql` provided in `/data/`. You will use it for every example in this chapter.

2.2 Day 2 – Your First SELECT

Day 2 The most useful 30 characters in tech.

```sql
SELECT  first_name, last_name, email      -- columns to return
FROM    customers                          -- source table
WHERE   country = 'India'                  -- row filter
ORDER BY signup_date DESC                  -- sort
LIMIT   10;                                -- only the first 10
```

Listing 2.1: *The anatomy of a SELECT*

2.2.1 Distinct, aliases, comments

- `SELECT DISTINCT country FROM customers;` returns unique values only.
- `SELECT price * 0.18 AS gst FROM products;` renames a column.
- `--` starts a single-line comment; `/* ... */` a block comment.

2.3 Day 3 – Filtering with WHERE

Day 3 Five operators that handle 95% of filters.

Operator	Example
=, <>, <, >, <=, >=	`price > 1000`
BETWEEN	`order_date BETWEEN '2026-01-01' AND '2026-03-31'`
IN	`country IN ('India','UAE','Singapore')`
LIKE	`email LIKE '%@gmail.com'`
IS NULL	`phone IS NULL`

Watch Out

`phone = NULL` is always false. Use `phone IS NULL`. This is the single most common SQL bug for beginners.

2.4 Day 4 – ORDER BY, GROUP BY, HAVING

Day 4 From rows to insights.

2.4.1 Sorting

`ORDER BY total_amount DESC, customer_id ASC` sorts by total amount descending, ties broken by customer id.

2.4.2 The five aggregates

`COUNT(*)`, `SUM(x)`, `AVG(x)`, `MIN(x)`, `MAX(x)`. They collapse many rows into one.

2.4.3 GROUP BY – the Pivot Table of SQL

```sql
SELECT   region,
         SUM(amount) AS revenue,
         COUNT(*)     AS n_orders,
         AVG(amount) AS avg_order_value
FROM     orders
WHERE    status = 'completed'
GROUP BY region
HAVING   SUM(amount) > 100000
ORDER BY revenue DESC;
```

Listing 2.2: *Revenue by region, only regions over 1 lakh*

Concept

WHERE filters rows *before* grouping. **HAVING** filters groups *after* aggregation. If you put an aggregate in `WHERE`, the database will refuse to run the query.

2.5 Day 5 – Functions That Clean Real Data

Day 5 Real-world data is messy. These functions fix it.

Family	Functions
String	`UPPER, LOWER, TRIM, LENGTH, SUBSTRING, CONCAT, REPLACE`
Number	`ROUND, CEIL, FLOOR, ABS, MOD, POWER`
Date	`NOW(), CURDATE(), DATE_FORMAT, DATEDIFF, DATE_ADD, EXTRACT`
Logic	`CASE WHEN ... THEN ... ELSE ... END`
Null	`COALESCE, IFNULL, NULLIF`
Convert	`CAST(x AS DECIMAL(10,2)), CONVERT(...)`

```sql
SELECT  customer_id,
        SUM(amount) AS lifetime_value,
        CASE
          WHEN SUM(amount) >= 50000 THEN 'platinum'
          WHEN SUM(amount) >= 10000 THEN 'gold'
          WHEN SUM(amount) >=  2000 THEN 'silver'
          ELSE 'bronze'
        END AS tier
FROM    orders
GROUP BY customer_id;
```

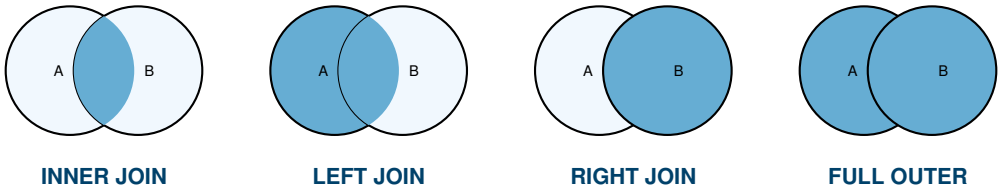
Listing 2.3: *Bucketing customers with CASE WHEN*

2.6 Day 6 – JOINs

Day 6 Putting tables back together.

Concept

A **JOIN** combines rows from two tables based on a matching column. **INNER** keeps only matches. **LEFT** keeps all rows from the left table even if there is no match (the right side becomes `NULL`). **RIGHT** is the mirror image. **FULL OUTER** keeps everything.



Shaded region = rows kept. Unshaded portions of A or B are dropped.

```sql
SELECT   c.name AS category,
         SUM(oi.quantity * oi.unit_price) AS revenue
FROM     order_items   oi
JOIN     products      p  ON p.product_id  = oi.product_id
JOIN     categories    c  ON c.category_id = p.category_id
JOIN     orders        o  ON o.order_id    = oi.order_id
WHERE    o.order_date BETWEEN '2026-05-01' AND '2026-05-31'
GROUP BY c.name
ORDER BY revenue DESC;
```

Listing 2.4: *Revenue per category in May 2026*

Try It Yourself

Write a single query that returns, for every customer in the `week2-ecommerce-database`: the customer's name, their total number of orders, and the date of their most recent order. Customers with zero orders should still appear. (Hint: `LEFT JOIN` from customers to orders.)

2.7 Day 7 – Project: E-commerce SQL Report

End-of-Week Project

Using the supplied e-commerce schema, write a `.sql` file containing the following queries, each with a one-line comment explaining its purpose:

1. Top 10 customers by lifetime revenue.
2. Monthly revenue trend for the last 12 months.
3. Category-level conversion rate (orders / unique visitors), if visit data is provided.
4. Customers who placed orders in 2024 but not in 2025 (churn candidates).
5. A single "executive summary" query that returns 8 KPIs in one row.

Submit the script plus a one-page memo with three insights and three recommendations.

Mistakes Beginners Make

1. `WHERE phone = NULL` – always false. Use `IS NULL`.
2. Writing an aggregate (`SUM`, `COUNT`) inside `WHERE` instead of `HAVING`. The database refuses; the error message is rarely clear.
3. Forgetting that `COUNT(col)` ignores nulls but `COUNT(*)` counts every row. They disagree silently.
4. Joining without an `ON` clause – Cartesian product on millions of rows. Your laptop fan will warn you.

5. Quoting column names with double quotes in MySQL when you meant backticks. `"order"` is interpreted as a string literal.
6. Mixing dialects: MySQL uses `DATE_FORMAT`, while PostgreSQL uses `TO_CHAR`. They are not interchangeable – check the docs before copy-pasting.

Interview Questions You'll Be Asked

- "What is the difference between `WHERE` and `HAVING`?"
- "Find the second-highest salary in an employees table."
- "Write a query that returns all customers who have never placed an order."
- "Explain the difference between `INNER`, `LEFT`, and `FULL OUTER` JOIN using a real example."
- "What happens to nulls in `COUNT(col)` versus `COUNT(*)`?"
- "Given an `orders` table, return the top three regions by revenue for the last completed month."

Key Takeaways

- Relational data lives in tables linked by keys.
- `SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ... LIMIT` is the canonical query shape.
- `WHERE` runs before grouping; `HAVING` runs after.
- Always use `IS NULL`, never `= NULL`.
- `INNER` for matches, `LEFT` when you must keep unmatched rows from the left table.

Up next: the analytical SQL that most interviews ask about – window functions, CTEs and subqueries.

CHAPTER 3

Advanced SQL & Analytics Queries

Week 3

Prerequisites: Chapter 2 – you can write `SELECT` with `JOIN`s and `GROUP BY`.
Estimated study time: 8–10 hours. Window functions and CTEs reward slow reading.
Tooling / dialect note: Examples primarily PostgreSQL; differences with MySQL are flagged where they appear.

OBJECTIVES

By the end of this week you will:

1. Write scalar, column and table subqueries (correlated and not).
2. Use `ROW_NUMBER, RANK, DENSE_RANK, LAG, LEAD, SUM() OVER()` for analytics.
3. Refactor nested queries into readable CTEs (`WITH ...`).
4. Build recursive CTEs for hierarchical data.
5. Read an `EXPLAIN` plan and recognise the obvious anti-patterns.

From the Instructor – Pawan

Most beginners struggle with window functions for the first 48 hours and then have an "oh, that's it?" moment around the third example. If you feel lost on Day 3, do not panic and do not skip ahead – close the book, sleep on it, and come back to the same example. The trick is realising that `PARTITION BY` is just "`GROUP BY` without collapsing the rows". Once that clicks, everything else lines up.

3.1 Day 1 – Subqueries

Day 1 A query inside another query.

Concept

A **subquery** is a SELECT statement embedded inside another statement. It can appear in the `SELECT`, `FROM` or `WHERE` clause and returns one of three shapes: a single value (scalar), a single column (list), or a full table (derived).

```sql
-- (1) Scalar: returns one value
SELECT name, price
FROM   products
WHERE  price > (SELECT AVG(price) FROM products);

-- (2) Column / list: returns one column, used with IN
SELECT *
FROM   customers
WHERE  customer_id IN (SELECT customer_id FROM orders
                       WHERE order_date >= '2026-01-01');

-- (3) Derived table: a full result used in FROM
SELECT region, avg_order
FROM (
  SELECT region, AVG(amount) AS avg_order
  FROM   orders
  GROUP BY region
) r
WHERE  avg_order > 5000;
```

Listing 3.1: *Three shapes of subquery*

3.2 Day 2 – Correlated Subqueries

Day 2 **One inner query per outer row – powerful but slow.**

```sql
SELECT  e.employee_id, e.name, e.salary, e.department
FROM    employees e
WHERE   e.salary > (
          SELECT AVG(e2.salary)
          FROM   employees e2
          WHERE  e2.department = e.department     -- correlation here
        );
```

Listing 3.2: *Each employee versus their own department*

Watch Out

A correlated subquery re-executes for every outer row. For datasets above a few hundred thousand rows, rewrite it as a window function (Day 3) or a JOIN with a pre-aggregated CTE – often 10–100× faster.

3.3 Day 3 – Window Functions, Part 1: Ranking

Day 3 **Aggregate, but keep every row.**

Concept

A **window function** computes an aggregate *over a window of rows related to the current row*, without collapsing the result like `GROUP BY` does. The window is defined by an `OVER` clause.

```sql
WITH ranked AS (
  SELECT  category_id,
          product_id,
          revenue,
          DENSE_RANK() OVER (
            PARTITION BY category_id
            ORDER BY     revenue DESC
          ) AS rnk
  FROM    product_revenue
)
SELECT * FROM ranked WHERE rnk <= 3;
```

Listing 3.3: *Top 3 products per category*

Three ranking functions, three behaviours when there is a tie:

- `ROW_NUMBER()` – 1, 2, 3, 4 (arbitrary tie-break).
- `RANK()` – 1, 2, 2, 4 (gaps after ties).
- `DENSE_RANK()` – 1, 2, 2, 3 (no gaps).

3.4 Day 4 – Window Functions, Part 2: Running & Comparing

Day 4 Running totals, moving averages, period-over-period.

```sql
SELECT  month_start,
        revenue,
        SUM(revenue) OVER (ORDER BY month_start) AS running_total,
        LAG(revenue, 1) OVER (ORDER BY month_start) AS prev_month,
        revenue - LAG(revenue, 1) OVER (ORDER BY month_start)
          AS mom_delta,
        AVG(revenue) OVER (ORDER BY month_start
          ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
          AS rolling_3m_avg
FROM    monthly_revenue;
```

Listing 3.4: *Running revenue and month-on-month growth*

Pro Tip

`ROWS BETWEEN ... PRECEDING AND CURRENT ROW` is the frame specification. It defines exactly which rows the window covers. Always be explicit – the default frame differs between databases.

3.5 Day 5 – Common Table Expressions (CTEs)

Day 5 Readable, reusable, debuggable.

Concept

A **CTE** (Common Table Expression) is a named, temporary result set declared with `WITH` and used in the query that follows. It does not persist beyond the statement.

```sql
WITH monthly AS (
    SELECT DATE_FORMAT(order_date,'%Y-%m-01') AS month_start,
           SUM(amount) AS revenue
    FROM   orders
    WHERE  status = 'completed'
    GROUP BY month_start
),
growth AS (
    SELECT month_start,
           revenue,
           LAG(revenue) OVER (ORDER BY month_start) AS prev_rev,
           (revenue - LAG(revenue) OVER (ORDER BY month_start))
             / NULLIF(LAG(revenue) OVER (ORDER BY month_start),0)
             AS mom_pct
    FROM   monthly
)
SELECT * FROM growth WHERE mom_pct < 0;
```

Listing 3.5: *Chained CTEs – one step per CTE*

3.5.1 Recursive CTEs

For hierarchies (org chart, category tree, bill-of-materials):

```sql
WITH RECURSIVE chain AS (
    SELECT employee_id, manager_id, name, 1 AS lvl
    FROM   employees
    WHERE  manager_id IS NULL                -- root
    UNION ALL
    SELECT e.employee_id, e.manager_id, e.name, c.lvl + 1
    FROM   employees e
    JOIN   chain c ON e.manager_id = c.employee_id
)
SELECT * FROM chain ORDER BY lvl, name;
```

Listing 3.6: *Walking a manager chain*

3.6 Day 6 – Query Optimisation, the Beginner's Edition

Day 6 If your query takes more than five seconds, read this.

3.6.1 The five anti-patterns

1. `SELECT *` on a 50-column table when you need three.
2. Functions on indexed columns (`WHERE YEAR(order_date)=2026` defeats the index; use a date range instead).
3. Correlated subqueries on large tables.
4. `OR` on different columns where each could be indexed separately – often rewriteable as `UNION ALL`.
5. Joining on a string when an integer key exists.

3.6.2 Reading EXPLAIN

Prepend `EXPLAIN` (MySQL) or `EXPLAIN ANALYZE` (PostgreSQL) to any query. Look for:

- `type=ALL` or `Seq Scan` – full table scan.
- `rows` much larger than what your final result returns.
- `Using filesort` on large data – consider an index.

Try It Yourself

On the e-commerce database, write a single query that returns – for each month of 2026 – (a) the total revenue, (b) the same month's revenue one year earlier, and (c) the year-over-year percentage change. Use a CTE for the monthly aggregation and `LAG()` with a 12-row offset for the comparison.

3.7 Day 7 – Project: Advanced Analytics Report

End-of-Week Project

Using a retail dataset of at least 100,000 rows, deliver a single `.sql` script that:

1. Segments customers into RFM (Recency, Frequency, Monetary) quintiles using window functions.
2. Builds a year-over-year revenue comparison with a single chained-CTE query.
3. Returns the top-3 products per category per month.
4. Identifies stockouts using `LAG` on daily inventory.
5. Includes `EXPLAIN` output for at least two queries, with before/after timings.

Mistakes Beginners Make

1. Using `RANK()` when you meant `ROW_NUMBER()` and then being surprised when ties produce two "first" rows.
2. Omitting the frame clause: `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` ver-

sus the default (which differs between MySQL and PostgreSQL). Always be explicit.

3. Putting a window function in `WHERE` – it must go in `SELECT` or in a wrapped CTE.
4. Forgetting that `LAG/LEAD` on the first/last partition row returns `NULL`. Cover with `COALESCE` when you need zero.
5. Trying to "optimise" before reading `EXPLAIN`. Trust measurement, not intuition.

Interview Questions You'll Be Asked

- "Difference between `ROW_NUMBER`, `RANK`, and `DENSE_RANK`? Give an example output."
- "Write a query that returns the top-3 products per category."
- "Compute month-on-month revenue growth using a CTE and `LAG`."
- "When would you choose a correlated subquery over a JOIN?"
- "Sessionise a clickstream: group consecutive events into sessions that end after 30 minutes of inactivity."

Key Takeaways

- Subqueries answer "compared to what?".
- Window functions aggregate without losing rows – the analyst's super-power.
- CTEs make complex SQL readable and debuggable; build queries step-by-step.
- Recursive CTEs handle hierarchies natively.
- Read `EXPLAIN`. Index the columns in your `WHERE` and `JOIN` clauses.

Up next: we leave the database and pick up the statistical foundations that distinguish a competent SQL writer from an analyst.

CHAPTER 4

Statistics Foundations for Data Analysis

Week 4

Prerequisites: Comfort with arithmetic. School algebra. No prior statistics assumed.

Estimated study time: 10–12 hours. This is the densest week. Pace yourself; do not binge.

Tooling / dialect note: R or Excel for the formulas; pen and paper for the worked examples. Drawings beat formulas at this stage.

OBJECTIVES

By the end of this week you will:

1. Compute and interpret mean, median, mode, variance, standard deviation, percentiles and IQR.
2. Identify skewness and kurtosis from a histogram.
3. Apply the empirical rule and compute Z-scores.
4. Use basic probability rules and Bayes' theorem.
5. Recognise the binomial, Poisson and normal distributions and pick the right one.
6. Understand sampling, the Central Limit Theorem, and confidence intervals.

From the Instructor – Pawan

Many non-tech learners find this the hardest week of the twelve. If you struggle, that is normal – it is not a sign you are in the wrong field. Read each definition twice, then *draw the distribution by hand* before you look at code. Statistics becomes intuitive only when you can sketch it. If after a day of effort an idea has not clicked, post your specific confusion in the cohort forum; it is almost always shared and fixable in five minutes.

4.1 Day 1 – Central Tendency

Day 1 Where is the middle of your data?

Three measures, three personalities.

Measure	Definition	Robust to outliers?
Mean	$\bar{x} = \frac{1}{n}\sum x_i$	No
Median	The 50th percentile	Yes
Mode	The most frequent value	N/A

Concept

Use the **median** for skewed data (income, house prices, transaction sizes). Use the **mean** for roughly symmetric data (heights, exam scores). When mean $\gg$ median, your distribution has a long right tail.

4.1.1 Variants worth knowing

- **Weighted mean**: each value has a weight, $\bar{x}_w = \frac{\sum w_i x_i}{\sum w_i}$. Used in GPAs, portfolio returns.
- **Trimmed mean**: drop the top/bottom 5%, then average. Compromise between mean and median.

4.2 Day 2 – Dispersion

Day 2 How spread out is your data?

$$\text{Variance:} \quad s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$\text{Standard deviation:} s = \sqrt{s^2}$$

$$\text{Coefficient of variation:} CV = \frac{s}{\bar{x}}$$

Compute these in three tools – they all give the same answer:

```r
# --- R ---
mean(salaries); median(salaries); sd(salaries); var(salaries)
quantile(salaries, c(0.25, 0.5, 0.75))
IQR(salaries)
summary(salaries)               # all-in-one

# --- Excel ---
# =AVERAGE(A2:A500)     =MEDIAN(A2:A500)    =STDEV.S(A2:A500)
# =VAR.S(A2:A500)       =QUARTILE.INC(A2:A500, 1)
# =QUARTILE.INC(A2:A500, 3) - QUARTILE.INC(A2:A500, 1)   # IQR

# --- SQL (PostgreSQL) ---
# SELECT AVG(salary), STDDEV_SAMP(salary), VAR_SAMP(salary),
#        PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY salary),
#        PERCENTILE_CONT(0.50) WITHIN GROUP (ORDER BY salary),
#        PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY salary)
# FROM employees;
```

Listing 4.1: *Descriptive statistics in R, Excel and SQL*

Pro Tip

Use the *sample* variance (denominator $n-1$) almost always. The n-denominator version is only correct when you really do have the whole population.

4.2.1 The five-number summary

Min, Q1, median, Q3, max. Plotted as a box-and-whisker, this single chart tells you centre, spread, skew, and outliers at a glance.

4.2.2 Detecting outliers with IQR

- $IQR = Q_3 - Q_1$.
- Outlier fence: below $Q_1 - 1.5 \cdot IQR$ or above $Q_3 + 1.5 \cdot IQR$.

4.3 Day 3 – Shape: Distributions

Day 3 What does the data "look like"?

4.3.1 Skewness and kurtosis

Skewness measures asymmetry. Positive skew = long right tail (income, delivery times). Negative skew = long left tail (exam scores when most pass). **Kurtosis** measures the heaviness of the tails. High kurtosis means more frequent extremes than a normal distribution predicts – a fact that has broken many a risk model.

4.3.2 The normal distribution

$X \sim N(\mu, \sigma^2)$ with bell-shaped PDF $f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$.

Concept

Empirical Rule (68-95-99.7). For any normal distribution, approximately 68% of values lie within $\pm 1\sigma$ of the mean, 95% within $\pm 2\sigma$, and 99.7% within $\pm 3\sigma$. This is the foundation of every quality-control chart you will ever see.

4.3.3 Z-scores: comparing apples to apples

$z = \frac{x-\mu}{\sigma}$. A z-score expresses how many standard deviations a value is from the mean. A new customer with z=2.3 on spending is in the top 1% of customers.

4.4 Day 4 – Probability Fundamentals

Day 4 The maths of uncertainty in two pages.

4.4.1 Three rules to memorise

$$\text{Addition: } P(A \cup B) = P(A) + P(B) - P(A \cap B)$$
$$\text{Multiplication: } P(A \cap B) = P(A) \cdot P(B \mid A)$$
$$\text{Bayes: } P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Try It Yourself

A diagnostic test has 95% **sensitivity** (it correctly identifies 95% of true positives) and 95% **specificity** (it correctly identifies 95% of true negatives). The disease occurs in 1% of the population – this is the **prevalence**. A random person tests positive. What is the probability they actually have the disease?

(Answer: about 16%. Bayes is the only way to see this. Walk through it on paper. Most learners are shocked at how low the answer is – that intuition gap is exactly why rare-disease screening is a real public health problem.)

4.5 Day 5 – Common Distributions

Day 5 Pick the right one and a lot of analysis writes itself.

Distribution	Type	When to use
Bernoulli	Discrete	A single yes/no trial (coin flip, click).
Binomial	Discrete	# successes in n independent yes/no trials.
Poisson	Discrete	Counts of rare events in fixed time/space.
Geometric	Discrete	Trials until the first success.
Uniform	Continuous	All values equally likely.
Normal	Continuous	Heights, errors, averages of many things.
Exponential	Continuous	Waiting times for Poisson events.

4.6 Day 6 – Sampling & Estimation

Day 6 Why a 1,000-person survey can speak for 1 billion.

Concept

Central Limit Theorem (CLT). The distribution of sample means approaches a normal distribution as the sample size grows, *regardless of the shape of the underlying population* (so long as the population has a finite variance). $n \geq 30$ is a rule of thumb; how well it holds depends on the original shape. Heavily skewed populations or those with extreme outliers need much larger n before the approximation is trustworthy.

4.6.1 Standard error

The standard deviation *of the sample mean*: $SE = \frac{s}{\sqrt{n}}$. Larger samples = smaller SE = tighter estimate.

4.6.2 Confidence intervals

A 95% CI for the mean is approximately $\bar{x} \pm 1.96 \cdot SE$. *Interpretation:* if we repeated the sampling many times, 95% of such intervals would contain the true population mean.

Watch Out

A 95% CI does *not* say "there is a 95% probability that the true mean is in this interval" – that is a Bayesian statement, not a frequentist one. Get this right in interviews.

4.6.3 Sampling methods

Simple random, **stratified** (force proportional representation per stratum), **cluster** (sample groups, then survey inside), **systematic** (every k-th unit). Convenience samples are not statistical samples.

4.7 Day 7 – Project: Statistical Analysis Report

End-of-Week Project

Using `week4-employee-salary.csv` (provided), deliver a one-page report that:

1. Reports mean, median, mode, SD, IQR for salary, overall and by department.
2. Identifies outliers using both the IQR rule and z-scores; cross-check.
3. Visualises the distribution and labels skewness and kurtosis.
4. Gives a 95% confidence interval for the mean salary in each department.
5. One paragraph: what would you ask the HR head, given these numbers?

Mistakes Beginners Make

1. Reporting the mean for income data, where the median is far more honest. The mean follows the long tail; the median follows the typical person.
2. Confusing standard deviation (spread of the data) with standard error (spread of the sample mean).
3. Quoting "95% confidence interval" as "95% probability the true mean is here". The frequentist CI says nothing of the sort.
4. Forgetting Bayes when reading test results – the diagnostic-test example bites real doctors, not just students.
5. Applying the empirical rule to data that is clearly not normal. Always plot a histogram first.

Interview Questions You'll Be Asked

- "When would you use the median instead of the mean?"
- "Explain the Central Limit Theorem to a non-statistician in two sentences."
- "What is the difference between standard deviation and standard error?"
- "Explain Bayes' theorem with a real-world example."
- "You're told 5% of users churned this month. Is that a statistically meaningful change? What would you need to know to answer?"

Key Takeaways

- Mean for symmetric, median for skewed.
- Variance and SD are the building blocks of every statistical test.
- Empirical rule: 68 / 95 / 99.7 for normal data.
- Bayes flips the conditional probability; learn it cold.
- CLT is why samples work; SE tells you how well.

Up next: from describing data to *testing claims* – hypothesis testing, correlation, and exploratory data analysis.

CHAPTER 5

Advanced Statistics & EDA

Week 5

Prerequisites: Chapter 4 – mean, SD, distributions, sampling.
Estimated study time: 8–10 hours. Most of the difficulty is conceptual, not computational.
Tooling / dialect note: R for the tests; any tool for EDA. Examples assume tidyverse.

OBJECTIVES

By the end of this week you will:

1. Formulate null and alternative hypotheses for a business problem.
2. Pick and run the right test (Z, t, χ^2, ANOVA).
3. Interpret a p-value correctly, including its limits.
4. Distinguish correlation from causation.
5. Perform end-to-end exploratory data analysis on a real dataset.
6. Design an A/B test, size it, and analyse the results.

From the Instructor – Pawan

A/B testing is the only part of statistics that you will use almost daily as an analyst in a product company. Spend extra time on Day 5. Two pitfalls I see again and again: stopping the test as soon as p-value dips below 0.05 ("peeking"), and forgetting to check that the random assignment actually produced balanced groups before reporting the result. Both have ruined careers; do not let them ruin yours.

5.1 Day 1 – Hypothesis Testing

Day 1 Turning a question into a verdict.

Concept

A **hypothesis test** answers: "Is the difference I see in the data *real*, or could it have happened by chance?". It does so by assuming a **null hypothesis** (no difference) and

asking how surprising the data would be if the null were true. The smaller the probability of seeing data this extreme under the null (the **p-value**), the more we doubt the null.

5.1.1 The five-step ritual

1. State H_0 (null) and H_1 (alternative).
2. Pick a significance level α, usually 0.05.
3. Compute the test statistic and the corresponding p-value.
4. If $p < \alpha$, reject H_0. Otherwise, fail to reject.
5. Report the result *with a confidence interval*, not just a p-value.

5.1.2 Two errors, two costs

Type I (α): convicting an innocent null – you declared an effect that wasn't real. **Type II** (β): acquitting a guilty null – you missed a real effect. **Power** $= 1 - \beta$.

Watch Out

$p = 0.04$ does *not* mean "there is a 4% chance the null is true." It means "if the null were true, we'd see data this extreme 4% of the time." The distinction matters in interviews and in court.

5.2 Day 2 – The Common Tests

Day 2 Pick the right tool for the data.

Test	Use it when ...
One-sample t	Compare a sample mean against a known value.
Two-sample t	Compare two independent group means.
Paired t	Same subjects before & after.
χ^2 (chi-square)	Two categorical variables – are they independent?
ANOVA (one-way)	Compare three or more group means.
Z-test	Like t-test but population SD known, large n.

```r
# Are the two groups' mean order values different?
t.test(amount ~ group, data = orders_ab, var.equal = FALSE)
# Welch's t is the default and safer than Student's
```

Listing 5.1: *Two-sample t-test in R*

5.3 Day 3 – Correlation, not Causation

Day 3 Just because two things move together ...

Concept

Pearson's r measures the *linear* association between two numeric variables, ranging from -1 to $+1$. **Spearman's** ρ is its rank-based cousin; it captures any *monotonic* relationship and is robust to outliers.

5.3.1 A tiny field guide

- $|r| < 0.3$: weak. 0.3–0.7: moderate. > 0.7: strong.
- $r = 0$ does not mean "no relationship"; only "no *linear* relationship". Always look at the scatter plot.
- A high correlation between A and B can be caused by a third variable C (*confounder*). Ice-cream sales correlate with drowning. Both depend on summer.

5.4 Day 4 – Exploratory Data Analysis

Day 4 The first hour with every new dataset.

EDA is the structured habit of *looking before you model*. There is no single recipe, but the following routine works on almost any tabular dataset.

1. **Shape**: rows, columns, types. `glimpse(df)` or `df.info()`.
2. **Missingness**: where, how much, why.
3. **Univariate**: histogram and boxplot of every numeric column; bar chart of every categorical column.
4. **Bivariate**: scatter for numeric–numeric; box plot for numeric–categorical; cross-tab + heatmap for categorical pairs.
5. **Correlation matrix** of the numeric columns.
6. **Outliers**: list the top/bottom 0.5% on every numeric variable; investigate the suspicious ones.
7. **Initial questions**: write three hypotheses to test next.

Pro Tip

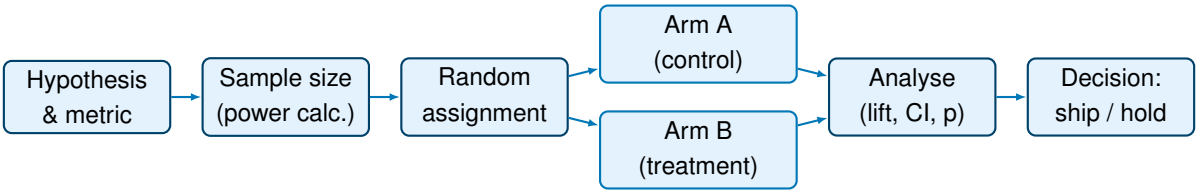
A useful rule: never go past the *shape*-and-*missingness* step without writing one sentence describing the dataset in your own words. If you can't, you are not ready to model it.

5.5 Day 5 – A/B Testing

Day 5 Hypothesis testing meets the business.

5.5.1 The setup

You have two versions of something – a webpage, an email, a button colour. You randomly assign each visitor to A (control) or B (treatment) and measure a conversion metric.



Run the whole pipeline once on synthetic data before you ever touch a live experiment. Bugs hide in steps 2 and 3.

5.5.2 Sample size

You need to know in advance how many users to test before stopping. A simple rule of thumb for a 5% absolute lift at 80% power and $\alpha = 0.05$ is roughly $n = 16 \cdot p(1-p)/d^2$ per arm, where p is baseline rate and d is the absolute detectable lift. Online calculators do this for you; use Evan Miller's or your statistics library.

5.5.3 Common mistakes

- Peeking at results early and stopping when the p-value dips below 0.05 – this is *p-hacking* and inflates Type I error.
- Running for too short a period and missing weekly cycles.
- Ignoring novelty effects – treatment lifts often decay.
- Sample-ratio mismatch: if A got 51% and B got 49%, your random assignment broke. Investigate before reporting.

5.6 Day 6 – Visualising for Analysis

Day 6 Charts that earn their pixels.

Goal	Chart
Distribution of one numeric var.	Histogram, density plot
Comparison of two distributions	Overlaid density, box plots
Trend over time	Line chart
Composition / parts of a whole	Stacked bar (avoid pie for > 3 slices)
Relationship of two numbers	Scatter plot with trend line
Geographic	Choropleth, point map

Watch Out

A few habits to drop:

- Truncated y-axes that exaggerate change.
- Pie charts with more than three slices.
- Rainbow colour scales for ordered data; use a sequential palette.
- 3D anything.

Try It Yourself

You ran an A/B test with 1,200 users per arm. Arm A converted 84 users, arm B converted 117. Compute: (1) the conversion rate for each arm, (2) the absolute and relative lift, (3) a two-proportion z-test statistic and its p-value. Decide whether to ship – and write the one sentence you would put in the experiment summary.

5.7 Day 7 – Project: EDA & Statistical Analysis

End-of-Week Project

Using `week5-marketing-ab-test.csv` (provided):

1. Run a full EDA: shape, missingness, univariate, bivariate plots.
2. State a hypothesis for the campaign's effect on conversion rate.
3. Run the appropriate test (two-proportion z-test or chi-square).
4. Report the lift, the p-value, and a 95% CI for the lift.
5. Write a one-page executive memo with a clear ship/hold recommendation.

Mistakes Beginners Make

1. Reporting only the p-value without the effect size or CI.
2. Peeking at A/B test results daily and stopping when "it looks significant".
3. Running a χ^2 test on a table with cell counts below 5 – the approximation breaks; use Fisher's exact test instead.
4. Treating a high correlation as causation without considering confounders.
5. Skipping the EDA routine and jumping straight to modelling. The worst bugs hide in step 1 (the data shape itself).

Interview Questions You'll Be Asked

- "What is a p-value – in plain English?"
- "Walk me through how you would design an A/B test for a new checkout button."
- "What is the difference between Type I and Type II error?"
- "You see a correlation of 0.8 between two variables. What would you check before claiming a relationship?"
- "Describe your usual EDA routine on a fresh dataset."

Key Takeaways

- Hypothesis testing answers "is this real or random?".
- Pick the right test based on data shape; report CI alongside p-value.
- Correlation $\neq$ causation. Investigate confounders.
- Follow a consistent EDA routine on every new dataset.
- Run A/B tests to a pre-committed plan; don't peek early.

Up next: a programming language that lets you do all of this – and build interactive dashboards – in one place.

CHAPTER 6

R Programming Fundamentals

Week 6

Prerequisites: No prior programming required. Statistics from Chapters 4–5 helps but is not strictly needed.

Estimated study time: 10–12 hours. Code along; do not just read.

Tooling / dialect note: R 4.x and RStudio Desktop (both free). Install the `tidyverse` package on Day 4.

OBJECTIVES

By the end of this week you will:

1. Install R and RStudio and write your first script.
2. Use R's core data types: vector, list, matrix, data frame, factor.
3. Write functions, conditionals and use the `apply` family.
4. Read data with `readr` and shape it with the `tidyverse` pipeline.
5. Apply `dplyr` verbs: `select, filter, mutate, arrange, summarise, group_by`.
6. Reshape data with `tidyr` and join multiple tables.

From the Instructor – Pawan

Before we begin: pick a habit. Either RStudio Projects from Day 1, or a single "scratchpad" folder you commit to git daily. The learners who treat code as a craft from the first week have, in my experience, noticeably cleaner capstones in Week 11. The ones who copy-paste into a Notepad and never source-control anything spend Week 11 untangling files they wrote a month earlier.

A second tip: do not skip the `tidyverse` pipe until you truly understand it. It looks like syntactic sugar, but it is the difference between code that reads like a recipe and code that reads like a puzzle.

6.1 Day 1 – R, RStudio and Hello World

Day 1 Setting up the tool you will use for the next six chapters.

Download R from `cran.r-project.org`, then RStudio Desktop (the free edition) from `posit.co`.

Launch RStudio; the four panes are **Script** (top-left), **Console** (bottom-left), **Environment / History** (top-right), **Files / Plots / Help** (bottom-right).

```r
# Arithmetic
2 + 2                  # [1] 4
log(100, base = 10)    # [1] 2

# Variables
revenue <- 1250000
cost    <- 870000
profit  <- revenue - cost
profit                 # [1] 380000

# Getting help
?mean
help.search("regression")
```

Listing 6.1: *A first R session*

Pro Tip

`<-` is R's preferred assignment operator (`Alt+-` in RStudio inserts it). `=` also works but is reserved by convention for function arguments.

6.2 Day 2 – Data Types and Structures

Day 2 The five shapes you'll see every day.

6.2.1 Atomic types

`numeric`, `integer`, `character`, `logical`, `complex`, `raw`. Check with `typeof(x)` or `class(x)`.

6.2.2 Vectors – the workhorse

```r
prices  <- c(199, 249, 89, 1299, 349)
in_sale <- prices < 300              # logical vector
prices[in_sale]                       # subset
mean(prices); sd(prices); range(prices)
```

6.2.3 Data frames – the table

```r
sales <- data.frame(
  product = c("A","B","C"),
  units   = c(10, 25, 7),
  price   = c(199, 249, 89)
)
sales$revenue <- sales$units * sales$price
sales[sales$revenue > 1000, ]
```

6.2.4 Lists and factors

A **list** can hold elements of different types and lengths – the JSON of R. A **factor** stores a categorical variable with a fixed set of levels and (optionally) an order; always use it for ordered categories.

6.3 Day 3 – Control Flow & Functions

Day 3 Make your code re-usable in 90 seconds.

```r
# Conditional
classify ← function(score) {
  if      (score >= 80) "A"
  else if (score >= 60) "B"
  else                  "C"
}

# Vectorised conditional -- usually preferred
ifelse(scores >= 60, "pass", "fail")

# For loop (rarely needed)
total ← 0
for (x in 1:10) total ← total + x

# apply family -- the R way
sapply(prices, function(p) p * 1.18)   # add 18% GST
```

Listing 6.2: *Conditionals, loops and functions*

Pro Tip

If you find yourself writing a `for` loop over rows of a data frame, stop. There is almost always a vectorised `dplyr` or `apply` version that is shorter and 10× faster.

6.4 Day 4 – The Tidyverse and the Pipe

Day 4 One philosophy, eight packages, infinite analyses.

Install once: `install.packages("tidyverse")`. Load: `library(tidyverse)`.

Concept

The **pipe** `%>%` (or the newer `|>`) takes the result on its left and passes it as the *first* argument of the function on its right. It turns nested function calls into a readable left-to-right recipe.

```r
# Nested -- read inside-out
head(arrange(filter(mtcars, hp > 100), desc(mpg)), 5)

# Piped -- read top-down
```

```r
mtcars %>%
  filter(hp > 100) %>%
  arrange(desc(mpg)) %>%
  head(5)
```

Listing 6.3: *Compare: nested vs piped*

6.4.1 Reading data

```r
library(readr)
orders <- read_csv("data/orders.csv")
write_csv(orders, "data/clean_orders.csv")
```

6.5 Day 5 – `dplyr`: Five Verbs Run Half Your Job

Day 5 select, filter, mutate, arrange, summarise.

```r
library(dplyr); library(lubridate)

orders %>%
  filter(order_date >= today() - days(90)) %>%
  inner_join(products, by = "product_id") %>%
  mutate(revenue = quantity * unit_price) %>%
  group_by(category) %>%
  summarise(
    n_orders = n(),
    revenue  = sum(revenue),
    avg_aov  = mean(revenue)
  ) %>%
  arrange(desc(revenue))
```

Listing 6.4: *Revenue by category in the last 90 days*

Verb	Does
`select()`	keep / drop columns
`filter()`	keep rows matching a condition
`mutate()`	create or modify columns
`arrange()`	sort rows
`summarise()`	collapse a group into one row
`group_by()`	set the grouping for subsequent verbs

6.6 Day 6 – Reshaping with `tidyr` & Joins

Day 6 Long, wide, and joined.

Concept

Tidy data: one row per observation, one column per variable, one table per kind of observational unit. Almost every painful analysis problem becomes easy once the data is tidy.

```r
# Wide → long
sales_long ← sales_wide %>%
  pivot_longer(cols = jan:dec, names_to = "month", values_to = "revenue")

# Long → wide
sales_wide_again ← sales_long %>%
  pivot_wider(names_from = month, values_from = revenue)

# separate & unite
df %>% separate(name, into = c("first","last"), sep = " ")
df %>% unite("date", year, month, day, sep = "-")
```

Listing 6.5: *pivot longer / wider*

6.6.1 Joins

The dplyr joins map directly to SQL: `inner_join()`, `left_join()`, `right_join()`, `full_join()`, `semi_join()` (keep matching rows), `anti_join()` (keep non-matching).

```r
orders %>%
  left_join(customers, by = "customer_id") %>%
  left_join(products,  by = "product_id")
```

Try It Yourself

Load the built-in dataset `mtcars`. Using one piped `dplyr` chain, return – per cylinder count – the count of cars, the mean `mpg`, and the heaviest car's model name. Hint: convert the rownames to a column first with `tibble::rownames_to_column()`.

6.7 Day 7 – Project: Data Wrangling Pipeline

End-of-Week Project

Build a single R script `wrangle.R` that:

1. Reads four CSV files (orders, items, customers, products) from `/data/week6/`.
2. Joins them into one tidy table.
3. Cleans column names with `janitor::clean_names()`.
4. Handles missing prices by imputing the category median.
5. Pivots to a monthly revenue table (one row per category, one column per month).

6. Writes the final tidy file to `data/analysis_ready.csv`.

Mistakes Beginners Make

1. Using = instead of <- for assignment. It works, but breaks tidyverse conventions and creates subtle bugs in function arguments.
2. Calling `filter()` from both `dplyr` and `stats`. Always load `dplyr` last, or explicitly `dplyr::filter()`.
3. Setting `stringsAsFactors=TRUE` accidentally (R $<$ 4.0 default). Use tibbles to avoid surprises.
4. Writing a `for` loop over rows of a data frame – there is almost always a vectorised alternative that is 10$\times$ faster.
5. Forgetting that `NA` is contagious. `sum(x)` returns `NA` if any value is `NA`; pass `na.rm = TRUE`.

Interview Questions You'll Be Asked

- "Difference between a list and a vector in R?"
- "What does the pipe operator do, and why is it useful?"
- "Write a single dplyr chain to compute revenue by month for the last 90 days."
- "How do you handle missing values? Walk me through your decision tree."
- "When would you reach for `purrr::map` instead of a loop?"

Key Takeaways

- Vectorise. Loops are rarely the answer.
- The pipe makes data analyses readable as recipes.
- Five `dplyr` verbs cover most everyday transformations.
- `tidyr` reshapes; `dplyr` computes; together they tame real-world data.

Up next: we turn these transformed tables into elegant visualisations – and write production-ready R.

CHAPTER 7

Advanced R & Data Visualisation

Week 7

Prerequisites: Chapter 6 – comfortable with `dplyr` verbs and the pipe.
Estimated study time: 10–12 hours. Most of the week is ggplot2; budget two days.
Tooling / dialect note: RStudio with tidyverse loaded. Optional: install `ggthemes` and `patchwork`.

OBJECTIVES

By the end of this week you will:

1. Build publication-quality charts with `ggplot2`.
2. Use facets, themes and custom palettes confidently.
3. Handle messy strings with `stringr` and regex.
4. Work with dates and times via `lubridate`.
5. Use advanced `dplyr`: `across()`, `case_when()`, window functions.
6. Write clean R scripts you would be happy to share with a teammate.

From the Instructor – Pawan

ggplot2 has a steep first hour and a gentle next hundred. If the layered grammar feels alien at first, that is the experience for every learner I have taught. Type out the seven canonical charts on Day 1 by hand, even if you understand what they do. The muscle memory pays back tenfold when you are debugging an interview chart under time pressure.

A quick aside on regex: the goal is not to memorise the syntax, it is to recognise a pattern when you see one. Bookmark `regex101.com` and let it explain anything you do not understand.

7.1 Day 1 – ggplot2 Fundamentals

Day 1 The grammar of graphics in one page.

Concept

A `ggplot2` plot is built in *layers*: a dataset, an aesthetic mapping (`aes()`: which column → which visual property), one or more geometric layers (`geom_*`), optional scales, facets, and a theme. Each `+` adds a layer.

```r
library(ggplot2)

ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()          # scatter
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_smooth()         # trend
ggplot(mpg, aes(x = class)) + geom_bar()                     # bar (count)
ggplot(mpg, aes(x = displ)) + geom_histogram(bins = 30)      # histogram
ggplot(mpg, aes(x = class, y = hwy)) + geom_boxplot()        # box
ggplot(economics, aes(x = date, y = unemploy)) + geom_line()# line
ggplot(diamonds, aes(carat, price)) + geom_density2d()       # density
```

Listing 7.1: *The seven canonical chart types*

7.1.1 Adding titles and labels – always

```r
ggplot(orders_by_month, aes(month, revenue)) +
  geom_col(fill = "#0077B6") +
  labs(title    = "Monthly revenue, 2026",
       subtitle = "Source: order_items",
       x = NULL, y = "Revenue (INR)",
       caption  = "Created with R + ggplot2") +
  theme_minimal(base_size = 12)
```

7.2 Day 2 – Faceting, Themes, Palettes

Day 2 Same chart, many lenses.

```r
ggplot(sales, aes(month, revenue, group = region)) +
  geom_line(colour = "#00416A", linewidth = 0.7) +
  facet_wrap(~ region, ncol = 3, scales = "free_y") +
  labs(title = "Revenue trend per region") +
  theme_minimal(base_size = 11) +
  theme(strip.background = element_rect(fill = "#E3F2FD", colour = NA),
        strip.text       = element_text(face = "bold"))
```

Listing 7.2: *Faceted small multiples*

7.2.1 Pick a palette on purpose

- Sequential (`Blues`) for ordered numeric data.
- Diverging (`RdBu`) for data with a meaningful zero.
- Qualitative (`Set2`) for unordered categories.

- viridis is colour-blind friendly and prints well in grayscale.

7.2.2 Saving with ggsave

```
ggsave("plots/revenue_trend.png", plot = last_plot(),
       width = 8, height = 5, dpi = 300)
```

7.3 Day 3 – Strings with stringr

Day 3 Cleaning the world's messy text data.

```
library(stringr)

str_to_upper("hello")              # "HELLO"
str_length("data analytics")       # 14
str_trim("  whitespace  ")         # "whitespace"
str_detect(emails, "@gmail\\.com") # logical vector
str_replace_all(phones, "[^0-9]", "")
str_extract(text, "[A-Z]{2,}")     # any all-caps token
str_split(name, " ", simplify = TRUE)
```

Listing 7.3: *A field guide*

7.3.1 Regex in 30 seconds

- . – any character. * – zero or more.
- \d – digit. \w – word char. \s – whitespace.
- [abc] – any of a, b, c. [^abc] – none of.
- ^ – start, $ – end. () – capture group.

Pro Tip

Build regex on regex101.com. It explains every token in plain English and saves hours.

7.4 Day 4 – Dates with lubridate

Day 4 Parsing and arithmetic that finally make sense.

```
library(lubridate)

ymd("2026-05-16")            # Date
dmy("16/05/2026")            # same
ymd_hms("2026-05-16 14:23:11")

today() + days(7)            # one week from now
```

```r
today() %--% ymd("2026-12-31") / months(1)    # months remaining

orders %>%
  mutate(
    month    = floor_date(order_date, "month"),
    weekday = wday(order_date, label = TRUE),
    is_weekend = wday(order_date) %in% c(1, 7)
  )
```

7.5 Day 5 – Advanced dplyr

Day 5 across(), case_when() and friends.

```r
# Apply same function to many columns
df %>% summarise(across(where(is.numeric), mean, na.rm = TRUE))

# Multi-branch logic
orders %>% mutate(
  tier = case_when(
    lifetime_value >= 50000 ~ "platinum",
    lifetime_value >= 10000 ~ "gold",
    lifetime_value >=  2000 ~ "silver",
    TRUE                    ~ "bronze"
  )
)

# Window functions in dplyr
orders %>%
  group_by(customer_id) %>%
  arrange(order_date) %>%
  mutate(order_rank    = row_number(),
         days_since_prev = as.numeric(order_date - lag(order_date)))
```

Listing 7.4: *The patterns you'll use forever*

7.6 Day 6 – Project Hygiene

Day 6 Code that won't embarrass you in six months.

7.6.1 A clean project layout

Folder	Contains
`data/`	raw and processed CSVs (gitignore the raw)
`R/`	helper functions (`utils.R`, `plots.R`)
`scripts/`	top-level pipelines: `01_clean.R`, `02_eda.R`, `03_model.R`
`output/`	generated plots and tables
`README.md`	how to run, in five lines or fewer

7.6.2 R Markdown / Quarto

For analyses meant to be read, write `.Rmd` or `.qmd` files that mix prose, code, and output. Render to HTML or PDF with one button.

Pro Tip

Use the `here::here()` function for paths and `renv` for package versions. Future you and your colleagues will thank you.

Try It Yourself

Using `economics` from `ggplot2`: build a faceted line chart with one panel per decade of the data, showing `unemploy` over time. Use `lubridate` to derive the decade column. Add a horizontal reference line at the overall mean. Save the result to `plots/unemployment.png` at 8x5 inches, 300 dpi.

7.7 Day 7 – Project: R Analytics Report

End-of-Week Project

Deliver a single `.Rmd` report on `week7-retail-sales.csv`:

1. Three faceted `ggplot2` charts of monthly revenue by region.
2. A correlation heat-map of numeric columns.
3. A clean-text section: extract product brands from a free-text `description` column using regex.
4. A time-series chart with a 7-day rolling mean.
5. A short executive summary and three recommendations.

The Rmd should knit to HTML without warnings.

Mistakes Beginners Make

1. Mapping a constant inside `aes()` – e.g. `aes(colour = "red")` – which creates a legend with one item labelled "red" instead of colouring the geom.
2. Forgetting `group=` on multi-line plots and getting a single zig-zag instead of one line per category.
3. Using a rainbow palette for ordered data; choose viridis or a sequential palette.
4. Treating dates as character. Always parse with `ymd()` or equivalent before plotting.
5. Writing 200 lines of regex when `stringr` has a one-liner (e.g. `str_extract_all`).

Interview Questions You'll Be Asked

- "Walk me through the layered grammar of ggplot2."
- "Show me a chart you are proud of and what design choices you made."
- "How would you facet a chart by region while sharing only the x-axis?"
- "Extract all phone numbers from a free-text comments column using regex."
- "Compute weekday vs weekend conversion rate over the last 60 days."

Key Takeaways

- Every `ggplot2` plot is `data + aes + geom + scale + facet + theme`.
- Always set a title, axis labels and a colour-blind-safe palette.
- `stringr` + regex unlock 90% of text-cleaning tasks.
- `lubridate` makes date arithmetic a one-liner.
- Organise projects the same way every time; future-you is paying.

Up next: we turn all of this into a deployed, interactive R Shiny dashboard – the project that anchors most analyst portfolios in this cohort.

CHAPTER 8

R Shiny – Interactive Dashboards

Week 8

Prerequisites: Chapters 6–7 – R, dplyr, ggplot2 are second nature.
Estimated study time: 10–12 hours, plus deployment time.
Tooling / dialect note: RStudio + the `shiny`, `shinydashboard`, `plotly`, `DT`, `bslib` packages; a free shinyapps.io account by Day 6.

OBJECTIVES

By the end of this week you will:

1. Build a Shiny app from a single file.
2. Understand reactivity: sources, conductors and endpoints.
3. Use the major input widgets and output renderers.
4. Lay out a dashboard with `shinydashboard` and `bslib`.
5. Embed interactive `plotly` charts and DT tables.
6. Deploy your app to `shinyapps.io`.

From the Instructor – Pawan

Shiny is the unique selling point of an R analyst – the one thing your Python-only competition cannot quickly replicate. A deployed Shiny dashboard with a public URL on your CV is genuinely rare and genuinely impressive in recruiter screens.

The conceptual hurdle this week is reactivity. It is not like a script you read top-to-bottom; it is closer to a spreadsheet, where changing a cell automatically updates everything that depended on it. Spend Day 3 on the diagram below until it feels obvious.

8.1 Day 1 – Hello, Shiny

Day 1 Your first interactive app in under 30 lines.

Concept

A Shiny app has two pieces: a `ui` that describes the page, and a `server` function that reads inputs and writes outputs. Shiny re-runs the server logic automatically whenever an input changes – this is **reactivity**, and it is the whole magic.

```r
library(shiny); library(ggplot2)

ui <- fluidPage(
  titlePanel("Hello Shiny"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins", "Number of bins:", 5, 50, 30)
    ),
    mainPanel(
      plotOutput("hist")
    )
  )
)

server <- function(input, output, session) {
  output$hist <- renderPlot({
    ggplot(faithful, aes(eruptions)) +
      geom_histogram(bins = input$bins,
                     fill = "#0077B6", colour = "white") +
      theme_minimal()
  })
}

shinyApp(ui, server)
```

Listing 8.1: *A complete one-file Shiny app*

Save as `app.R` and click *Run App* in RStudio.

8.2 Day 2 – UI Components and Layout

Day 2 The vocabulary of dashboards.

8.2.1 Inputs

`textInput()`, `numericInput()`, `sliderInput()`, `selectInput()`, `checkboxInput()`, `checkboxGroupInput()`, `radioButtons()`, `dateInput()`, `dateRangeInput()`, `fileInput()`, `actionButton()`.

8.2.2 Outputs

`plotOutput()` – `renderPlot()`.
`tableOutput()` – `renderTable()`.
`DT::DTOutput()` – `DT::renderDT()`.

`textOutput()` – `renderText()`.
`uiOutput()` – `renderUI()` (dynamic UI).

8.2.3 Layouts

`fluidPage()`, `fixedPage()`, `sidebarLayout()`, `fluidRow()` + `column()`, `tabsetPanel()`, `navbarPage()`.

8.3 Day 3 – Reactivity Deep Dive

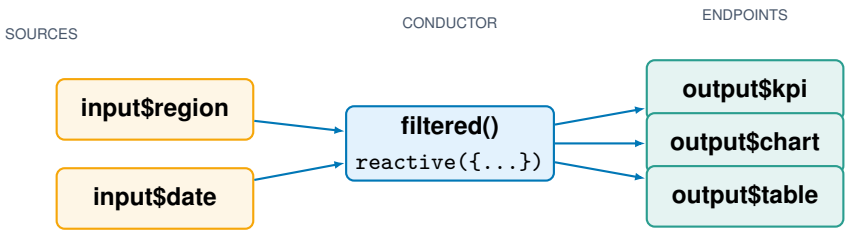
Day 3 The mental model you must own.

Concept

Reactivity has three roles:

- **Reactive sources** – inputs (`input$x`).
- **Reactive conductors** – `...` `reactive( )` expressions that depend on sources and feed endpoints.
- **Reactive endpoints** – outputs and observers (`output$plot`, `observe({})`).

Shiny lazily re-evaluates conductors only when their inputs change, and only when something downstream actually needs them.



The conductor `filtered()` computes once per input change, and all three endpoints read its cached value – no repeated filtering.

```
server ← function(input, output, session) {

  filtered ← reactive({
    orders %>%
      filter(region == input$region,
             order_date %within% input$dr[1] %--% input$dr[2])
  })

  output$kpi    ← renderText(sum(filtered()$amount))
  output$chart  ← renderPlot(ggplot(filtered(), aes(...)) + ...)
  output$table  ← DT::renderDT(filtered())
}
```

Listing 8.2: *Cache an expensive filter with reactive()*

8.3.1 When to use what

- `reactive()`: computes *a value* that other code reads.
- `observe()`: runs for side effects (writes a file, updates a widget).
- `observeEvent(input$btn, {})`: explicit, run on button click.
- `eventReactive(input$btn, {})`: computed value, refreshed only when the button fires.

Watch Out

Calling a reactive without parentheses (`filtered`) returns the function itself, not the value. Always `filtered()`. This is a top-three rookie bug.

8.4 Day 4 – Interactive Visualisations

Day 4 Beyond static plots.

```
library(plotly); library(DT)

ui ← fluidPage(
  plotlyOutput("scatter"),
  DTOutput("tbl")
)

server ← function(input, output, session) {

  output$scatter ← renderPlotly({
    plot_ly(mtcars, x = ~wt, y = ~mpg, type = "scatter",
            mode = "markers", color = ~factor(cyl)) %>%
      layout(title = "Weight vs MPG")
  })

  output$tbl ← renderDT({
    datatable(mtcars, filter = "top",
              options = list(pageLength = 10))
  })
}
```

Listing 8.3: *Plotly + DT*

Click-, hover- and brush-events from `plotly` can be read back into the server via `event_data("plotly_click")`, letting you build true drill-down interfaces.

8.5 Day 5 – shinydashboard & bslib

Day 5 Professional-looking UI without writing CSS.

```
library(shinydashboard)

```

```r
ui ← dashboardPage(
  dashboardHeader(title = "Sales Cockpit"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Overview",  tabName = "overview",  icon = icon("home")),
      menuItem("Customers", tabName = "customers", icon = icon("user")),
      menuItem("Products",  tabName = "products",  icon = icon("box"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem("overview",
        fluidRow(
          valueBoxOutput("rev"),
          valueBoxOutput("orders"),
          valueBoxOutput("aov")
        ),
        fluidRow(box(plotOutput("trend"), width = 12))
      ),
      tabItem("customers", DTOutput("cust_tbl")),
      tabItem("products",  plotOutput("prod_chart"))
    )
  )
)
```

Listing 8.4: *A KPI dashboard skeleton*

8.5.1 Modern themes with `bslib`

`bs_theme(version = 5, bootswatch = "minty")` drops a polished Bootstrap-5 theme into any Shiny app. Combine with `cards()` and `layout_columns()` from `bslib` for an entirely custom-CSS-free dashboard.

8.6 Day 6 – Deployment

Day 6 From localhost to a public URL.

8.6.1 shinyapps.io in five clicks

1. Create a free account at `shinyapps.io`.
2. In RStudio: `Tools > Global Options > Publishing > Connect`.
3. Open your `app.R`, click the blue *Publish* icon.
4. Pick the files to deploy. RStudio installs everything in the cloud.
5. Open the URL it gives you. Done.

8.6.2 What to do before deploying

- Move large data into `.rds` files (much smaller than CSV).

- Pin package versions with `renv` so the cloud build matches your laptop.
- Wrap risky operations in `tryCatch()` so a bad input doesn't crash the whole app.
- Add a `showNotification()` for slow operations.

Pro Tip

The free tier gives you five apps and 25 hours of compute / month. Plenty for a portfolio piece. For heavier loads, deploy with Docker on a VPS.

Try It Yourself

Extend the Day 1 *Hello Shiny* app: add a `selectInput` to choose between `eruptions` and `waiting` (both columns of the `faithful` dataset), and a `checkboxInput` to toggle a density overlay on the histogram. Wrap the data selection in a `reactive()` so both outputs share one evaluation.

8.7 Day 7 – Project: Interactive Business Dashboard

End-of-Week Project

Deploy a multi-tab Shiny dashboard for the e-commerce dataset:

1. Tab 1: KPI overview (revenue, AOV, orders, conversion) with date range filter.
2. Tab 2: Region drill-down – click a region to filter the product chart below.
3. Tab 3: Customer table (`DT`) with column filters and CSV export.
4. Tab 4: Plotly scatter – units vs revenue, sized by margin, coloured by category.
5. Deployed to `shinyapps.io` with a public URL.

Submit: GitHub repo link + live URL.

Mistakes Beginners Make

1. Calling a reactive without parentheses (`filtered` instead of `filtered()`). Top-three rookie bug; the error is rarely obvious.
2. Using `observe({ })` for things that should be `reactive({ })`, then watching the app re-run side effects.
3. Loading large CSVs at every reactive call instead of once at app startup.
4. Not pinning package versions with `renv`; the cloud build breaks because your laptop has dplyr 1.2 and shinyapps.io installs 1.1.
5. Forgetting to wrap risky inputs in `validate(need(...))`; one weird user click crashes the app for everyone.

Interview Questions You'll Be Asked

- "Explain reactivity in Shiny to a colleague who knows R but not Shiny."
- "Difference between `reactive()`, `observe()`, and `eventReactive()`?"
- "How do you make a Shiny app faster without rewriting it?"
- "When would you pick Shiny over Power BI for an internal tool?"
- "Walk me through deploying a Shiny app to a public URL."

Key Takeaways

- Shiny = ui + server + reactivity.
- Cache expensive computations in `reactive()`.
- `plotly` and `DT` add interaction with no extra HTML.
- `shinydashboard` / `bslib` make apps look professional, fast.
- Deployment to `shinyapps.io` is five clicks; do it for every portfolio project.

Up next: the BI tools the rest of the industry uses, so you know when to reach for them and when R Shiny is the better answer.

CHAPTER 9

Power BI Fundamentals (Optional Track)

Week 9

Prerequisites: Chapter 2 (SQL basics) helps but is not required.

Estimated study time: 6–8 hours. Optional but recommended – most BI roles list it.

Tooling / dialect note: Power BI Desktop (free, Windows only). On macOS, install via a Windows VM or skip to Chapter 10 (Tableau).

OBJECTIVES

By the end of this week you will:

1. Navigate the three views of Power BI Desktop: Report, Data, Model.
2. Connect to Excel, CSV, SQL and web data sources.
3. Shape data in Power Query with applied steps.
4. Build a clean star-schema data model.
5. Write basic DAX measures, including time intelligence.
6. Understand row context vs filter context (the heart of DAX).
7. Publish reports to the Power BI Service.

From the Instructor – Pawan

Power BI is the de-facto standard in Microsoft-stack enterprises – mostly finance, healthcare, retail and operations roles in India and the Middle East. Even if you prefer R or Tableau, learning Power BI doubles the number of roles you can credibly apply to.

The single concept that separates beginners from pros is **row context vs filter context** in DAX. We cover it on Day 6; do not skim that section.

9.1 Day 1 – The Power BI Stack

Day 1 Three apps, one workflow.

- **Power BI Desktop** – the free Windows authoring tool. This is where you do almost all the work.

- **Power BI Service** – the cloud workspace where you publish, share and schedule refreshes.
- **Power BI Mobile** – read-only apps for iOS/Android.

Concept

Power BI is built on three engines: **Power Query** (for shaping data), the **VertiPaq** in-memory columnar database, and **DAX** (for modelling and measures). Knowing which tool to reach for in each engine is half the skill.

9.1.1 When to use Power BI vs R Shiny

Use Power BI when ...	Use R Shiny when ...
You need fast, standardised business reports for many users.	You need bespoke logic, custom models or non-tabular interactions.
Your stakeholders already live in Microsoft 365.	Your reports are research-driven or need to embed statistical models.
You want row-level security with AAD groups.	You want full control of the UI and code.

9.2 Day 2 – Connecting to Data

Day 2 Get Data is where every project starts.

`Home > Get Data` brings up a catalogue of 100+ connectors: Excel, CSV, SQL Server, MySQL, PostgreSQL, SharePoint, REST APIs, and hundreds more. Choose your mode at import time:

- **Import** – data is cached in the model. Fastest queries; scheduled refresh required.
- **DirectQuery** – queries run against the source on every interaction. Always live, but slower and limited DAX.
- **Live Connection** – only for Analysis Services / PBI datasets.

9.3 Day 3 – Power Query

Day 3 The data-cleaning workhorse.

Click *Transform Data* to open the **Power Query Editor**. Every action you take here is recorded as a step in the *Applied Steps* pane. This is the M language behind the scenes – but you rarely need to write it by hand.

9.3.1 The ten transformations you'll use every day

1. Promote first row to headers.
2. Change column types.
3. Filter rows; remove blanks and duplicates.
4. Replace values; trim and clean text.
5. Split column by delimiter.

6. Pivot / Unpivot columns.
7. Merge queries (left/inner/anti join).
8. Append queries (union).
9. Add custom column with an M expression.
10. Group by with an aggregation.

Pro Tip

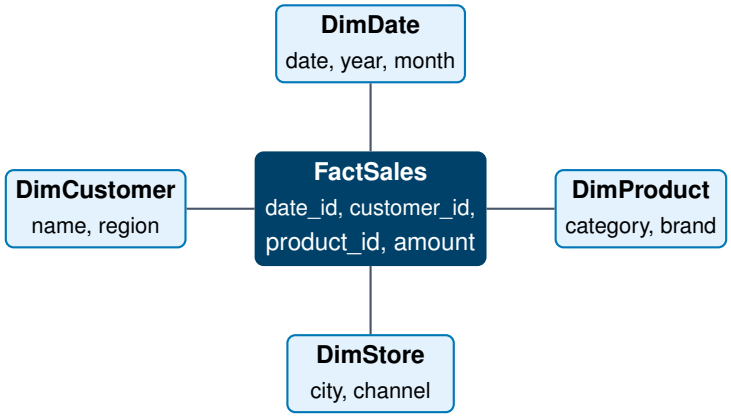
Do as much shaping as possible in Power Query, not in DAX. PQ runs once at refresh; DAX runs on every visual interaction.

9.4 Day 4 – Data Modelling

Day 4 Star schemas are not optional.

Concept

A **star schema** has one or more **fact tables** (orders, sales, events – big, narrow, additive) joined to several **dimension tables** (date, customer, product – small, descriptive). Power BI's engine is optimised for this shape. Wide "flat" tables are an anti-pattern.



Fact in the centre, dimensions around. Every relationship is many-to-one from fact → dimension.

9.4.1 Relationships

Drag a column from one table to another in the Model view. Set:

- **Cardinality**: usually *Many to One* – many fact rows per dimension row.
- **Cross-filter direction**: *Single* by default. *Both* is rarely needed and easy to misuse.

9.4.2 A date table – always create one

```
DateTable =
ADDCOLUMNS(
```

```
  CALENDAR(DATE(2020,1,1), DATE(2030,12,31)),
  "Year",        YEAR([Date]),
  "MonthNum",    MONTH([Date]),
  "MonthName",   FORMAT([Date], "mmm"),
  "Quarter",     "Q" & FORMAT(QUARTER([Date]), "0"),
  "YearMonth",   FORMAT([Date], "yyyy-mm")
)
```

Listing 9.1: *DAX: a calendar table*

Mark it as a date table (`Modeling > Mark as date table`) and relate your fact's date column to it.

9.5 Day 5 – Visualisations & Reports

Day 5 One canvas, many pages.

9.5.1 Picking the right visual

Power BI's visualisation pane has 30+ built-in visuals. The same rules apply as in chapter 5: line for trends, bar for ranks, scatter for relationships. **Avoid** 3D charts, dual-axis gauges and rainbow heatmaps.

9.5.2 Slicers, filters, drill-through

- **Slicer** – a visible filter widget.
- **Filter pane** – finer-grained, off to the side.
- **Drill-through** – click a category in a chart, jump to a detail page filtered to that value.

9.6 Day 6 – DAX Fundamentals

Day 6 The language of business measures.

Concept

DAX (Data Analysis Expressions) is the formula language used to define *calculated columns* (per-row) and *measures* (aggregated on demand). Almost always, you want a measure: it adapts to whatever filter context the visual provides.

```
Total Revenue   := SUM(Sales[Amount])

YTD Revenue     := TOTALYTD([Total Revenue], DateTable[Date])

Revenue PY      := CALCULATE([Total Revenue],
                     SAMEPERIODLASTYEAR(DateTable[Date]))

YoY %           := DIVIDE([Total Revenue] - [Revenue PY],
```

```
                            [Revenue PY])

Top 5 Customers := CALCULATE(
                      [Total Revenue],
                      TOPN(5, VALUES(Customer[Name]), [Total Revenue])
                    )
```

Listing 9.2: *Five DAX measures you'll re-use forever*

9.6.1 CALCULATE – the keystone

`CALCULATE(expr, filter1, filter2, ...)` evaluates `expr` in a modified filter context. Mastering CALCULATE is the difference between a Power BI beginner and a pro.

9.6.2 Row context vs filter context

Concept

Row context is what a calculated column sees: one row at a time. **Filter context** is what a measure sees: the set of filters active in the current visual (slicers, row labels, column labels, page filters, etc.). `CALCULATE` is the only way to *modify* filter context. `SUMX/AVERAGEX` iterate over rows – they create row context inside a measure.

Two practical implications:

- A measure that returns the same number in every visual cell almost always means the filter context is being lost – usually because `ALL()` or `REMOVEFILTERS()` was used too aggressively.
- A calculated column that references another table without using `RELATED()` fails because there is no row context to follow across the relationship.

9.6.3 Relationship traps

- **Inactive relationships**: turn one on inside CALCULATE with `USERELATIONSHIP()`. Useful for "ship date" vs "order date" on the same DateTable.
- **Bidirectional filtering**: convenient, occasionally necessary, often a footgun. Defaults should be single-direction.
- **Ambiguity**: more than one path between tables. Power BI refuses to pick; restructure your model.

Try It Yourself

You have a measure `[Total Revenue]` on a star-schema model with a proper `DateTable`. Write three DAX measures: (1) Revenue in the last 7 days, (2) Revenue same period last year, (3) Year-over-year % growth. Then put all three on a card visual filtered by current month.

9.7 Day 7 – Project: Power BI Sales Dashboard

End-of-Week Project

Using the e-commerce dataset:

1. Connect Orders, Items, Customers, Products tables (use the provided `.pbix` starter).
2. Build a star schema with relationships.
3. Create a date table and time-intelligence measures.
4. Build a three-page report: *Overview, Customers, Products.*
5. Add slicers for region, category and date range.
6. Publish to the Power BI Service; share the link.

Mistakes Beginners Make

1. Building wide flat tables and complaining the model is slow. Always star-schema first.
2. Using calculated columns where a measure would do. Columns bloat the model; measures stay in memory only when needed.
3. Turning on bidirectional filtering "just in case". This single click can break filter context across the entire report.
4. Skipping the dedicated date table and using the fact table's `Order Date` directly – time intelligence then fails silently.
5. Writing all logic in the visual "Filters on this visual" pane. Two weeks later, nobody can find why a number is wrong.

Interview Questions You'll Be Asked

- "Difference between row context and filter context in DAX?"
- "When would you use a calculated column vs a measure?"
- "What does CALCULATE do? Walk me through an example."
- "Why do we always create a separate date table?"
- "What is the risk of bidirectional filtering?"

Key Takeaways

- Shape in Power Query, model in DAX, visualise in the report canvas.
- Star schema beats one flat table every time.
- Always create a proper date table.
- Measures > calculated columns. Both > visual-level formulas.
- CALCULATE is the spine of advanced DAX. Row context vs filter context is the unlock for every advanced pattern.

Up next: the third tool in your BI portfolio – Tableau.

CHAPTER 10

Tableau Essentials (Optional Track)

Week 10

Prerequisites: Comfort reading a star-schema model (Chapter 9 covers it).
Estimated study time: 6–8 hours. Optional but recommended if the role lists Tableau.
Tooling / dialect note: Tableau Public (free, Windows and macOS).

OBJECTIVES

By the end of this week you will:

1. Connect to data and navigate Tableau Desktop.
2. Build the seven canonical chart types in Tableau.
3. Use the Marks card – colour, size, shape, label, detail, tooltip.
4. Create calculated fields, parameters and table calculations.
5. Use level-of-detail (LOD) expressions: `FIXED`, `INCLUDE`, `EXCLUDE`.
6. Understand extracts vs live connections, and blending vs joins.
7. Assemble worksheets into an interactive dashboard with filter actions.
8. Publish to Tableau Public.

From the Instructor – Pawan

If you already know Power BI, you will pick up Tableau in two days; the hardest part is unlearning DAX habits. The single feature interviewers ask about most often is LOD expressions – I have added a section in Day 4 you should not skip.

10.1 Day 1 – Tableau, in 10 Minutes

Day 1 The vocabulary you must know on Day 1.

Term	Means
Data source	Connection to a file or database.
Worksheet	One view (one chart).
Dashboard	A canvas combining many worksheets.
Story	A sequence of dashboards.
Dimension	A descriptive field (usually categorical).
Measure	A numeric field that you aggregate.
Marks card	The panel that maps fields to visual properties.
Shelves	Rows / Columns / Filters / Pages.

Pro Tip

Tableau Public is free, with the catch that all your published work is publicly visible. For portfolio pieces, this is a feature.

10.2 Day 2 – Basic Visualisations

Day 2 Drag, drop, chart.

The flow is always: drag a dimension to Rows, a measure to Columns (or vice versa), and Tableau picks a sensible default. The **Show Me** panel suggests the chart families compatible with your selection.

Charts in this order:

- Bar / Column – ranks and comparisons.
- Line – trends over time.
- Scatter – relationships of two measures.
- Map – geographic, with auto-detected lat/long.
- Bullet, KPI, Big-Number – summaries.

10.3 Day 3 – Advanced Visuals

Day 3 Beyond the Show Me defaults.

- **Dual-axis**: drop a second measure on the right Y-axis, then right-click "Synchronise Axis".
- **Combination charts**: one measure as bars, another as line.
- **Treemaps / heat maps**: hierarchical compositional data.
- **Reference lines / bands**: averages, targets, control bands.
- **Trend lines**: linear, polynomial, logarithmic; check R-squared in tooltip.

10.4 Day 4 – Calculated Fields & Parameters

Day 4 Logic without leaving Tableau.

Concept

A **calculated field** is a new column computed from existing ones; you write its formula in a Tableau expression. A **parameter** is a single, user-controlled value that any calculation can read – a perfect way to add "what-if" sliders.

10.4.1 The five formula families

- String: `UPPER`, `TRIM`, `CONTAINS`, `SPLIT`.
- Date: `DATEDIFF`, `DATETRUNC`, `TODAY`.
- Logical: `IF/ELSEIF`, `CASE WHEN`, `IIF`.
- Aggregate: `SUM`, `AVG`, `COUNTD`.
- Table calc: `RUNNING_SUM`, `WINDOW_AVG`, `INDEX`.

```
// Customer tier
IF [Total Revenue] >= 50000 THEN "Platinum"
ELSEIF [Total Revenue] >= 10000 THEN "Gold"
ELSEIF [Total Revenue] >=  2000 THEN "Silver"
ELSE "Bronze"
END

// % of total within partition (table calc)
SUM([Sales]) / TOTAL(SUM([Sales]))
```

Listing 10.1: *Examples of calculated fields*

10.4.2 Level-of-detail (LOD) expressions

Concept

LOD expressions let you compute aggregates at a different granularity than the visual you are looking at.

- **FIXED**: ignore the visual filters and compute at the named dimension. Use for "customer lifetime value" shown next to a date-filtered chart.
- **INCLUDE**: add a dimension to the visual's level. Use for "average of per-region totals" on a national chart.
- **EXCLUDE**: remove a dimension from the visual's level. Use for "share of category" on a per-product table.

```
{ FIXED [Customer ID] : SUM([Sales]) }            // lifetime value
{ INCLUDE [Region]    : AVG([Sales]) }            // avg per-region within group
```

```
{ EXCLUDE [Category]  : SUM([Sales]) }          // total ignoring category
```

Listing 10.2: *LOD examples*

10.4.3 Extracts vs live, blending vs joining

- **Live connection**: queries the source on every interaction. Always fresh, slower, dependent on the source's load.
- **Extract (`.hyper`)** : snapshot pulled into Tableau's engine. Much faster; refresh on a schedule.
- **Joining**: at the data-source level, returns a single wide table. Fast, but inflates row counts on many-to-many keys.
- **Blending**: at the visual level, treats one data source as "primary" and looks up aggregates from a "secondary". Useful for keeping fact tables at different granularities separate, but slower and limited in calculations.

10.5 Day 5 – Dashboards

Day 5 Worksheets in a single, opinionated canvas.

10.5.1 Build flow

1. Each chart goes on its own worksheet first; polish there.
2. File > New Dashboard. Pick a device-aware size.
3. Drag worksheets, filters, and parameters onto the canvas.
4. Use **Containers** (Horizontal / Vertical) for clean alignment.
5. Add **Filter Actions**: clicking a chart filters another.

10.5.2 Five design rules

- One clear question per dashboard; one "hero" chart on top.
- Maximum five colours; reserve red/green for delta.
- White space matters. Crowded dashboards hide insight.
- Tooltips should answer the next likely question.
- Always test in a fresh user's eyes – ideally on a phone.

10.6 Day 6 – Publishing & Comparing Tools

Day 6 Three tools, one career.

10.6.1 Tableau Public

`Server > Tableau Public > Save to Tableau Public`. The workbook gets a public URL you can pin to your portfolio.

10.6.2 Tableau vs Power BI vs R Shiny

Tool	Best at
Tableau	Visual exploration, beautiful defaults, story telling.
Power BI	Microsoft 365 ecosystems, enterprise governance, DAX measures.
R Shiny	Custom logic, statistical models, fully bespoke UI.

Pro Tip

The interview question is rarely "which is best?" but "why did you pick this one?". Knowing the trade-offs cold makes you a credible analyst.

Try It Yourself

Using Tableau's built-in *Sample - Superstore* dataset: build a dashboard with (a) a bar chart of sales by Category, (b) a line chart of sales over time, (c) a map of sales by State. Add a filter action so that clicking a Category bar filters both the line chart and the map.

10.7 Day 7 – Project: Published Tableau Dashboard

End-of-Week Project

Re-create the e-commerce dashboard from chapter 8, but in Tableau:

1. Three worksheets: revenue trend, region map, top-products bar.
2. One dashboard with cross-filtering between all three.
3. Two calculated fields (customer tier, YoY %).
4. One parameter – a "target revenue" that drives a reference line.
5. Published to Tableau Public; URL added to your portfolio.

Write a one-paragraph comparison with your Shiny and Power BI versions.

Mistakes Beginners Make

1. Confusing `FIXED` and `INCLUDE`: `FIXED` ignores visual filters; `INCLUDE` respects them. The two give very different numbers.
2. Joining when you should have blended (or vice versa). If your row counts mysteriously exploded, you joined two facts at different grains.
3. Forgetting that data-source filters are not visual filters; `FIXED` still ignores the latter but not the

former.

4. Using "Show Me" for everything and never learning the Marks card. Interviewers can spot it.
5. Publishing a dashboard with a default sheet name ("Sheet 1") or without a tooltip strategy. Polish matters.

Interview Questions You'll Be Asked

- "Difference between a calculated field, a table calculation, and an LOD expression?"
- "When would you use `FIXED` vs `INCLUDE`?"
- "Extract vs live connection – how do you choose?"
- "Walk me through Tableau's order of operations (filters → marks)."
- "Show me a Tableau dashboard you have built. Why those chart choices?"

Key Takeaways

- Dimensions go on rows/columns; measures get aggregated.
- Calculated fields and parameters cover most "custom logic" needs.
- LOD expressions free you from the visual's level of detail.
- Filter actions are what make Tableau dashboards *feel* interactive.
- Always finish a project with a published, public URL.

Up next: two weeks to put everything together – your end-to-end capstone project and the career skills to apply for your first analyst role.

CHAPTER 11

Capstone Project – Part 1

Week 11

Prerequisites: Everything from Chapters 1–10. A working Git/GitHub habit (set one up if you haven't – see the mini-section on Day 2).

Estimated study time: Full week. The capstone is the longest single piece of work in the course.

Tooling / dialect note: Whatever stack you have used: SQL + R + Shiny is the default; Python + Streamlit + DuckDB is an acceptable substitute.

OBJECTIVES

By the end of this week you will:

1. Choose a domain and scope an analytics project end-to-end.
2. Source real-world data ethically and reproducibly.
3. Build a SQL + R analysis pipeline for the project.
4. Wireframe and prototype an R Shiny dashboard.
5. Present a Week 11 checkpoint review for mentor feedback.

From the Instructor – Pawan

Your capstone matters more than every previous week combined when it comes to landing an interview. Hiring managers skim CVs in 30 seconds; a live dashboard URL stops them. Pick a project you find genuinely interesting – this matters because the next two weeks will be hard, and intrinsic motivation is what gets people across the finish line.

A small request: do not pick a project that requires data you do not have. "I'll scrape it" is a common reason capstones never ship. If your dream project needs data you cannot get, downscope to a related public dataset and keep the bigger idea for after you have a job.

11.1 Day 1 – Choose Your Project

Day 1 The biggest single decision of the course.

A capstone is not graded on technical complexity alone. Recruiters look for: *a clear question, a defensible method, a polished output, and a useful conclusion.* Pick a project where you have access to data and a real audience – yourself or a friend's business counts.

Concept

A capstone is *not* a portfolio of unrelated charts. It is a single end-to-end story: one business question → one dataset → one method → one deployed artefact → one decision. Every section of your README, every slide of your presentation, and every tab of your Shiny app must pull in the same direction.

11.1.1 Starter ideas, by domain

Domain	Projects
Retail / E-commerce	Sales forecast, customer segmentation, inventory dashboard.
Finance / BFSI	Stock screener, credit-risk app, expense tracker.
Healthcare / Pharma	Clinical-trial KPI tracker, hospital-resource planner, disease-trend explorer.
Marketing	Campaign attribution, A/B test viewer, content-performance dashboard.
Operations	SLA monitor, defect Pareto, fleet utilisation.

11.1.2 The project proposal template

1. **Problem statement** (one paragraph).
2. **Audience** – who would use this, and what decision would they make?
3. **Data sources** – exactly which datasets, where they come from, licensing.
4. **KPIs** – the three to seven numbers your dashboard must compute.
5. **Deliverables** – a deployed Shiny app, GitHub repo, 2-page memo, 5-minute video walkthrough.
6. **Timeline** – daily milestones for Weeks 11–12.
7. **Risks** – what could go wrong, and your mitigation.

11.2 Day 2 – Sourcing Data & Setting Up Git

Day 2 Five places real datasets actually live – plus the version control your future self will thank you for.

11.2.1 Git in 10 minutes (the bare minimum)

If you have not used Git yet, do this once and forget about it for the rest of the week:

```
# one-time setup
git config --global user.name  "Your Name"
git config --global user.email "you@example.com"

# start the repo
cd capstone/
git init
git add .
git commit -m "Initial project skeleton"
```

```
10
11 # connect to GitHub
12 git remote add origin https://github.com/<you>/capstone.git
13 git push -u origin main
14
15 # daily loop
16 git status                    # see what changed
17 git add R/mod_overview.R      # stage specific files
18 git commit -m "Add overview tab with KPI cards"
19 git push
```

Listing 11.1: *The eight commands that cover 95% of capstone work*

Pro Tip

Add a `.gitignore` on day one. Never commit `data/raw/`, `.env` files, API keys, large `.rds` files, or RStudio's `.Rproj.user` folder. `github.com/github/gitignore` has ready-made templates.

- **Kaggle Datasets** – well-documented; pick ones with a clear licence.
- **Government open data** – `data.gov.in`, `data.gov.uk`, `data.gov`.
- **UCI Machine Learning Repository** – classics with good provenance.
- **Public APIs** – World Bank, OpenWeather, Spotify, TwitterX (paid tier).
- **Web scraping** – only when the site's ToS allow and you respect `robots.txt`.

Watch Out

Before importing a dataset, read its licence. If you can't redistribute it, include a clear note in your README and link to the original instead of committing the file.

11.3 Day 3 – Data Preparation & SQL Pipeline

Day 3 From raw files to a queryable database.

11.3.1 A cleaning checklist

- Standardise column names (`snake_case`).
- Parse dates and numbers.
- Handle missing values explicitly (drop? impute? flag?).
- De-duplicate.
- Document every choice in a `NOTES.md`.

11.3.2 Load into SQL

```sql
CREATE TABLE staging_orders (
  order_id     INT,
  customer_id  INT,
  order_date   DATE,
  amount       DECIMAL(12,2),
  region       VARCHAR(64),
  status       VARCHAR(16)
);

LOAD DATA LOCAL INFILE 'data/orders.csv'
  INTO TABLE staging_orders
  FIELDS TERMINATED BY ','
  ENCLOSED BY '"'
  IGNORE 1 LINES;

CREATE VIEW v_orders_completed AS
  SELECT * FROM staging_orders WHERE status = 'completed';
```

Listing 11.2: *A reproducible loader*

11.3.3 Five analytical views to build first

1. Daily / monthly revenue.
2. Customer lifetime value.
3. Top-N products per category.
4. Cohort retention (sign-up month × months since).
5. Funnel: visit → add-to-cart → purchase.

11.4 Day 4 – Statistical Analysis in R

Day 4 Stress-test your conclusions.

Run the EDA recipe from chapter 5 on the cleaned data. For every KPI you plan to show:

- Plot its distribution and check for outliers.
- Check for seasonal and weekday effects.
- Test the most interesting comparison with a t-test or chi-square – and report the confidence interval.
- Document at least one finding that surprised you.

Pro Tip

Interviewers love surprising findings. "Tuesday is our biggest revenue day, not Friday, and here is why" is a far better story than "revenue went up".

11.5 Day 5 – Shiny App, Part 1 – Skeleton

Day 5 Build the bones before the muscles.

11.5.1 Wireframe first

On paper or in any tool, sketch each tab: header, filter sidebar, KPI strip, primary chart, secondary chart, data table. Get a stakeholder to sign off before writing UI code.

11.5.2 Project layout

```
capstone/
  app.R              # entry point (or split into ui.R / server.R)
  R/
    mod_overview.R   # one Shiny module per tab
    mod_customers.R
    mod_products.R
    plots.R          # ggplot helpers
    queries.R        # SQL / dplyr helpers
  data/
    raw/             # immutable
    processed/       # built by scripts/
  scripts/
    01_load.R
    02_clean.R
    03_export_rds.R
  www/
    style.css
    logo.png
  renv.lock
```

11.5.3 Modules: the right way to grow a Shiny app

For any app over 150 lines, break each tab into a Shiny module (`moduleServer()` + UI function). Modules namespace their inputs, prevent collisions, and let you re-use the same component multiple times.

11.6 Day 6 – Shiny App, Part 2 – Polish

Day 6 Make it production-grade.

11.6.1 Performance

- Pre-aggregate slow queries into `.rds` files; load once at app start.
- Use `bindCache()` to memoise expensive reactives.
- Profile with `...  profvis::profvis( )`.

11.6.2 Error handling and UX

- `validate(need(input$x, "Pick a region"))` to gate plots.
- Wrap risky code in `tryCatch` with a friendly error toast.
- Add a loading spinner with `shinycssloaders::withSpinner()`.
- Provide a `downloadButton()` on every table.

11.6.3 Style

Pick one theme via `bslib::bs_theme()` and never look back. Use a consistent palette, generous white-space, and a clear primary action per page.

Try It Yourself

Before Day 7's checkpoint, do a peer-walkthrough rehearsal: present your capstone in exactly 5 minutes to a friend who is *not* a data analyst. Time yourself. The point of failure is almost always slide 2 – your problem statement is jargon-heavy or the audience does not see why the question matters. Re-write it three different ways before the checkpoint.

11.7 Day 7 – Checkpoint Review

Day 7 Submit, present, get feedback.

End-of-Week Project

Week 11 Checkpoint Deliverables

1. Project proposal (1 page).
2. Cleaned data + load scripts in GitHub repo.
3. Five SQL views with documentation.
4. EDA notebook with at least one surprising insight.
5. Working Shiny prototype (need not be deployed yet).
6. 5-minute walkthrough video (Loom or screen-recorded).

Schedule a 30-minute review with your mentor. Bring questions, not defensiveness.

Mistakes Beginners Make

1. Picking a project that needs data you do not have. By Day 5 you run out of options and start the week over.
2. Skipping the proposal step. Without a one-page plan, scope creep is guaranteed.
3. Committing the raw data to GitHub. Slow clones, sometimes legal issues. Use `.gitignore`.

4. Building all UI before any working analytics. The dashboard looks ready, but the numbers are wrong, and you cannot debug the layered code.
5. Hiding bad findings. "Nothing changed" is a legitimate result if your method is sound.

Interview Questions You'll Be Asked

- "Walk me through your capstone – what was the business question?"
- "How did you handle missing data? What assumptions did you make?"
- "What is one finding that surprised you?"
- "What would you do differently if you had another month?"
- "Show me your GitHub commit history. How do you commit?"

Key Takeaways

- Scope drives everything; pick a project you can finish, not the most impressive idea.
- Document data sources and cleaning choices; reproducibility is a differentiator.
- Build the Shiny app in modules from day one.
- Hunt for one counter-intuitive insight – that's your story.
- Commit early, commit often; your GitHub history is part of your portfolio.

Up next: ship it, write the README, polish your CV, and prepare for your first analyst interview.

CHAPTER 12

Capstone Completion & Career Prep

Week 12

Prerequisites: A working capstone prototype from Week 11.
Estimated study time: Full week. Plan, do not procrastinate.
Tooling / dialect note: A LinkedIn account; a public GitHub; `shinyapps.io` (or equivalent) account.

OBJECTIVES

By the end of this week you will:

1. Deploy your capstone Shiny app to a public URL.
2. Write a professional README and portfolio entry.
3. Optimise your CV and LinkedIn for analyst roles.
4. Practise the technical and behavioural interview rituals.
5. Deliver a polished 15-minute capstone presentation.

From the Instructor – Pawan

A word on expectations. This workbook will not, by itself, get you a job; the application process is hard, and luck plays a real role. What it will do is make sure you are *prepared* – so that when an opportunity appears, you do not waste it. Be patient with yourself. Most of my learners take 1–4 months after Week 12 to land their first analyst role. The ones who keep applying, keep practising SQL, and keep adding to their portfolio are the ones who eventually get there.

A final non-technical tip: be honest in interviews. Companies hire people they want to work with, not people who can recite definitions. A confident "I haven't used that yet, but here's how I would learn it" is almost always better than a half-bluff.

12.1 Day 1 – Ship It

Day 1 Deployment day.

12.1.1 Deployment checklist

- All R scripts re-run from a fresh checkout without manual steps.
- `renv` (or a `DESCRIPTION` file) pins package versions.
- Secrets (API keys, DB passwords) are read from environment variables, not committed.
- App opens cleanly on a phone-size screen.
- `showNotification("Loading...")` for any slow operation.
- Footer credits you (name, GitHub URL, LinkedIn).

12.1.2 Deploy in five clicks

Open RStudio, open `app.R`, click the blue Publish icon, pick **shinyapps.io**, choose the files, click Publish. Your app gets a URL like `your-name.shinyapps.io/capstone`.

Pro Tip

If your free tier runs out of compute, you can self-host with `shiny-server` on a $5 VPS, or push a Docker image to `Render`, `Fly.io` or `Posit Connect Cloud`.

12.2 Day 2 – Documentation & Portfolio

Day 2 If it's not on GitHub, it doesn't exist.

12.2.1 The README that lands interviews

A great repo README has exactly these sections, in order:

1. **Title** + one-sentence pitch.
2. Screenshot or animated GIF of the live app.
3. Live demo link.
4. Problem statement: what business question does this answer?
5. Data sources & licence.
6. Methods (a short paragraph, not a textbook).
7. Key findings – 3 bullet points.
8. How to run locally (5 commands max).
9. Folder structure.
10. License + your contact.

12.2.2 GitHub profile README

Make your profile `github.com/<you>/<you>` repo and put a one-page README there. Include: pinned projects, a short bio, skills, contact links, and what you're learning now.

12.3 Day 3 – Resume & LinkedIn

Day 3 Make the first 10 seconds count.

12.3.1 A one-page resume skeleton you can paste-and-edit

```
YOUR NAME
Aspiring Data Analyst | SQL . R Shiny . Power BI . Excel
City, Country | +91 XXXXXXXXXX | your.email@example.com
linkedin.com/in/your-handle | github.com/your-handle

SUMMARY
Data analyst trained at EDUSHARK with hands-on capstone in SQL,
R Shiny and Power BI. Built and deployed a live analytics
dashboard. Seeking junior analyst role in BFSI / Healthcare.

SKILLS
SQL (MySQL, PostgreSQL); R (tidyverse, ggplot2); R Shiny;
Power BI (DAX, Power Query); Tableau; Excel; Python (basic); Git.

PROJECTS
- Retail Sales Cockpit  -  R Shiny + SQL  -  [LIVE URL]
  120k-row dataset. RFM segmentation. Top decile = 38% of revenue.
  Deployed via renv on shinyapps.io; load time < 2s.
- E-commerce SQL Analysis  -  PostgreSQL  -  [GITHUB]
  15 analytical queries; window functions, CTEs; 8x speedup via
  index tuning documented with EXPLAIN.

CERTIFICATIONS
EDUSHARK TRAINING - Applied Data Analytics & BI (YYYY-MM)

EDUCATION
Degree, University, YYYY-YYYY
```

Listing 12.1: *Plain-text resume template (ATS-friendly)*

12.3.2 The fresher data-analyst resume, in one page

1. **Header**: name, role you want, email, phone, GitHub, LinkedIn, location.
2. **Summary** (2 lines): "Data analyst trained at EDUSHARK with hands-on projects in SQL, R Shiny and Power BI. Looking for an analyst role in BFSI / Healthcare / Retail."
3. **Skills**: SQL, R, Tidyverse, R Shiny, Power BI, Tableau, Excel, Python (basics), Git.
4. **Projects** (the heart of the resume, 3–4 entries). Each: title → tech → outcome with a number.
5. **Certifications** – EDUSHARK, plus any free Coursera/Datacamp.
6. **Education** – last because projects matter more.

Watch Out

ATS killers to avoid: tables (many scanners can't read them), icons-as-images, two-column PDFs from Canva, photos. Plain one-column ATS-friendly templates from `flowcv.com` or `rxresu.me` win.

12.3.3 LinkedIn checklist

- Professional photo, brand-coloured banner.
- Headline: "Aspiring Data Analyst | SQL, R Shiny, Power BI".
- About: same pitch as the resume summary, two paragraphs.
- Featured: pin the capstone live URL and GitHub.
- Skills + endorsements: add the 10 you used in this course.
- Open to Work badge *on* (recruiter-visible).

12.4 Day 4 – Technical Interviews

Day 4 The drills that matter.

12.4.1 The five buckets

1. **SQL live coding**: window functions, CTEs, second-highest salary, sessionisation.
2. **Statistics**: explain p-value, CLT, when to use median.
3. **Case studies**: "revenue dropped 10% – diagnose."
4. **Tool comparison**: "why R Shiny over Power BI here?"
5. **Your project**: walk me through the capstone.

12.4.2 A practice plan

- LeetCode SQL – aim for 60 problems across Easy + Medium.
- Stratascratch / Datalemur for case-style questions.
- Khan Academy stats refresher.
- Pair-mock with a peer once a week.

12.5 Day 5 – Behavioural Interviews

Day 5 The hardest hour you'll prep for.

Concept

The **STAR** framework: answer behavioural questions in four beats – **S**ituation, **T**ask, **A**ction, **R**esult. The result must be quantified.

12.5.1 Prepare seven stories, total

Cover: leadership, conflict, failure, learning quickly, handling ambiguity, persuading without authority, beating a deadline. One strong story per theme is enough; the same story can answer two questions.

12.5.2 The "tell me about yourself" opener

Three minutes, three beats: *Where I am now* (course, recent projects), *how I got here* (background + motivation), *where I want to go* (the role I'm interviewing for).

12.5.3 Questions you should ask

- What does success look like at six months?
- Walk me through a recent analysis the team ran end-to-end.
- What's the team's tooling – SQL, BI, scripting languages?
- How is data quality maintained?

12.6 Day 6 – Presentation Practice

Day 6 15 minutes that decide everything.

12.6.1 Slide outline

1. Title + your name + role you want.
2. The business question (one sentence).
3. The data and method (one slide each, max).
4. Key findings – three slides, one finding each.
5. Live demo of the Shiny app (2–3 minutes).
6. Recommendations – three actionable bullets.
7. Q&A prep slide (hidden) with anticipated questions.

12.6.2 Five delivery tips

- Open with the punchline, not the methodology.
- Rehearse the demo on the actual screen-share setup once.
- One number per slide, big, with context.
- Pause after questions; do not fill silence.

- End with a confident, written next-step.

Try It Yourself

Record a 90-second video answer to "Tell me about yourself." Watch it back with the sound off the first time – check your body language and eye contact. Then with sound – check pacing, filler words and clarity. Three takes is enough. Save the best one; you will use it as the introduction in your first screening call.

12.7 Day 7 – Graduation

Day 7 You did it.

End-of-Week Project

Final Capstone Presentation

- 15-minute live presentation to a panel.
- Live demo of deployed R Shiny dashboard.
- Q&A with peers and instructors.
- Certificate of completion.

Grading

- Capstone project quality – 40%
- Presentation & communication – 30%
- Documentation & portfolio – 20%
- Program participation – 10%

Mistakes Beginners Make

1. A two-column "Canva" CV that ATS scanners cannot read.
2. A GitHub README without screenshots or a live URL. Recruiters do not clone repos.
3. Memorising STAR answers. They sound rehearsed; tell stories instead.
4. Filling silence after a question. Pause – it is a sign of thinking, not weakness.
5. Applying to jobs without targeting the CV. A one-line tweak per application beats sending the same PDF to a hundred companies.

Interview Questions You'll Be Asked

- "Tell me about yourself." (3 minutes, three beats: now, past, future.)

- "Tell me about a time you handled disagreement with a stakeholder."
- "What is the most important thing you learned during your capstone?"
- "Why this company?" (do your homework.)
- "Where do you see yourself in three years?"

Key Takeaways

- Shipping is the only thing that counts in a portfolio.
- READMEs and screenshots get more eyes than code does.
- ATS-friendly CVs beat beautiful ones.
- Practise SQL live, statistics out loud, and project narration in front of a mirror.
- Behavioural answers in STAR; quantify every result.

Congratulations. You have completed twelve weeks of training in Data Analytics, SQL, R Programming, R Shiny and BI tools. You have a deployed dashboard, a public portfolio, and – most importantly – the habits of an analyst.

Be patient. Keep applying. I am rooting for you.

— Pawan

Chapter A

Glossary

A/B test
A randomised controlled experiment comparing two variants of something to estimate the causal effect of a change.

Aggregate function
A function that collapses a set of rows into a single value (`SUM`, `AVG`, `COUNT`, `MIN`, `MAX`, `MEDIAN`).

ANOVA
Analysis of Variance: a test for differences in means across three or more groups.

ATS
Applicant Tracking System – software recruiters use to screen CVs.

Bayes' theorem
The rule for updating a probability after observing new evidence: $P(A|B) = P(B|A)P(A)/P(B)$.

BFSI
Banking, Financial Services and Insurance – one of the largest hiring sectors for analysts.

CTE
Common Table Expression – a named, temporary result set in SQL declared via `WITH`.

CLT
Central Limit Theorem – the distribution of sample means approaches normality regardless of the population shape.

Cardinality
The numeric relationship between two related tables (1:1, 1:many, many:many).

Choropleth
A map shaded by a quantitative value per region.

Confidence interval
A range that, under repeated sampling, would contain the true parameter a specified percentage of the time.

Correlation
A measure of the linear (Pearson) or monotonic (Spearman) association between two variables.

DAX
Data Analysis Expressions – the formula language of Power BI.

Descriptive analytics
Analytics that answer "what happened?".

Diagnostic analytics
Analytics that answer "why did it happen?".

Dimension table
A descriptive table joined to a fact table in a star schema.

EDA
Exploratory Data Analysis – the structured habit of understanding a dataset before modelling.

Fact table
A table of measurable events (orders, transactions) joined to dimension tables.

Filter context
In DAX, the set of filters active when a measure is evaluated.

GROUP BY
SQL clause that collapses rows into groups for aggregation.

Hypothesis test
A procedure for deciding whether data is statistically consistent with a null hypothesis.

Imputation
Replacing missing values with a plausible substitute (median, mean, model output).

INDEX-MATCH
The classical Excel two-step lookup that survives column reordering, unlike `VLOOKUP`.

IQR
Interquartile range, $Q_3 - Q_1$. A robust measure of spread.

Join
An operation combining rows from two tables based on a related column.

KPI
Key Performance Indicator – a metric tied to a goal with a target and an owner.

Lagging indicator
A metric measuring outcomes that have already occurred.

Leading indicator
A metric measuring inputs that predict future outcomes.

Measure (DAX)
An aggregated calculation evaluated at query time in Power BI.

Median
The middle value of an ordered dataset; the 50th percentile.

Normal distribution
A symmetric bell-shaped distribution defined by its mean and standard deviation.

Null
In SQL, a marker for unknown / missing values. Always tested with `IS NULL`.

OVER()
The SQL clause that turns an aggregate into a window function.

Outlier
A data point unusually far from the rest of the distribution.

Pivot table
A two-dimensional summary table in Excel; SQL equivalent: `GROUP BY`.

Power Query
The data-shaping engine in Power BI; uses the M language.

Predictive analytics
Analytics that answer "what will happen?".

Prescriptive analytics
Analytics that answer "what should we do?".

Primary key
A column or combination that uniquely identifies a row.

P-value
The probability of observing data at least as extreme as the sample, assuming the null hypothesis is true.

R Shiny
An R framework for building interactive web applications and dashboards.

Reactivity
Shiny's automatic re-evaluation model: outputs update whenever their input dependencies change.

Regex
Regular expression – a pattern language for matching text.

RFM
Recency, Frequency, Monetary – a classic customer segmentation framework.

Sampling
Selecting a subset of a population to estimate its properties.

SMART
Specific, Measurable, Action-oriented, Relevant, Time-bound – a framework for analytical questions.

Standard deviation
The square root of the variance; spread in the data's units.

STAR
: Situation, Task, Action, Result – the framework for behavioural interview answers.

Star schema
: A data-model pattern with fact tables surrounded by dimension tables; the canonical BI shape.

Stringr
: A Tidyverse package for consistent string manipulation in R.

Subquery
: A SELECT statement embedded inside another SQL statement.

Tibble
: A modern R data frame from the Tidyverse with cleaner printing and stricter type behaviour.

Tidy data
: One row per observation, one column per variable.

Type I error
: Rejecting a true null hypothesis (false positive).

Type II error
: Failing to reject a false null hypothesis (false negative).

VLOOKUP
: A vertical lookup function in Excel; superseded by `XLOOKUP`.

Window function
: A SQL function that computes across a set of rows related to the current row without collapsing them.

XLOOKUP
: The modern Excel lookup function that handles errors, horizontal and vertical lookups in one call.

Z-score
: Number of standard deviations a value lies from the mean.

CHAPTER B

Further Resources

B.1 Free books worth your time

- Hadley Wickham & Garrett Grolemund, *R for Data Science* (2nd ed.) – the canonical tidyverse book. `r4ds.hadley.nz`.
- Hadley Wickham, *Mastering Shiny*. `mastering-shiny.org`.
- Cole Nussbaumer Knaflic, *Storytelling with Data*.
- Joe Celko, *SQL for Smarties* – the classical reference.

B.2 Online practice

- `leetcode.com/problemset/database` – SQL drills.
- `stratascratch.com` – analytics case-style SQL.
- `datalemur.com` – SQL interview questions.
- `kaggle.com/learn` – short, well-paced micro-courses.

B.3 Cheat-sheets and references

- Posit / RStudio Cheatsheets – `posit.co/resources/cheatsheets/`.
- Microsoft's DAX guide – `dax.guide`.
- Power BI documentation – `learn.microsoft.com/power-bi`.
- Tableau Help – `help.tableau.com`.

B.4 Communities

- `stackoverflow.com` (tags: `r`, `sql`, `shiny`, `powerbi`).
- RStudio Community – `community.rstudio.com`.
- r/dataisbeautiful, r/dataengineering on Reddit.
- Tableau Public gallery – inspiration and recipes.

B.5 Datasets to keep practising on

- Kaggle Datasets (`kaggle.com/datasets`).
- UCI ML Repository.
- Indian government open data – `data.gov.in`.
- World Bank Open Data – `data.worldbank.org`.
- Our World in Data – `ourworldindata.org`.
- NYC Open Data – `opendata.cityofnewyork.us`.

B.6 Newsletters & podcasts

- "Data Elixir" weekly newsletter.
- "The Analytics Engineering Roundup".
- "Towards Data Science" on Medium (filter for quality).
- Podcasts: *Data Skeptic*, *Super Data Science*.

B.7 Where to host your portfolio

- `github.io` – the free option, infinite control.
- `shinyapps.io` – free tier for live R Shiny apps.
- `tableau.com/public` – Tableau dashboards.
- `public.powerbi.com` – shareable Power BI reports.
- `quarto.org/docs/publishing` – multi-format publishing.

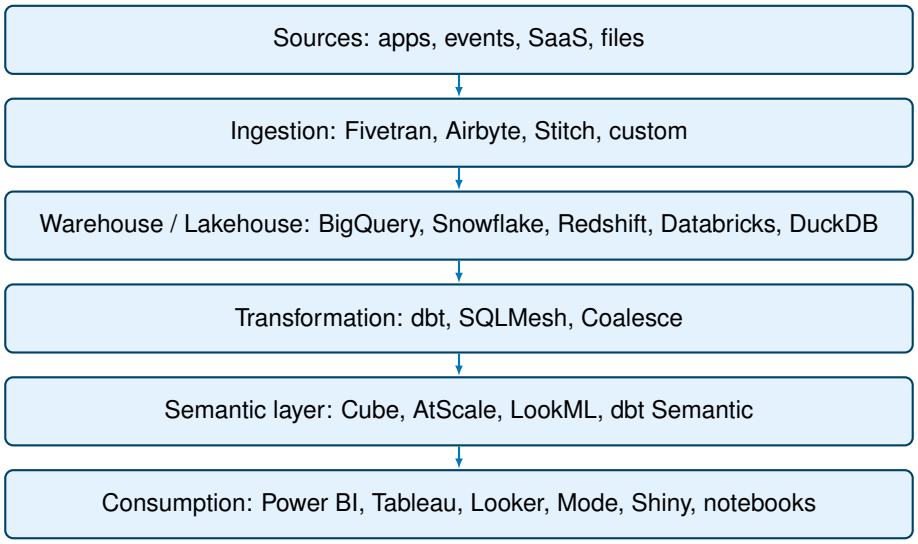
Bookmark a few, ignore the rest, and ship something this week.

CHAPTER C

The Modern Analytics Landscape

The main book teaches you the canonical tools: SQL, R, ggplot2, Shiny, Power BI, Tableau. They will carry you through almost any analyst role. This appendix is a short tour of what the rest of the industry is adopting in 2026 – not because you must learn them all, but because you will encounter them in job descriptions, podcasts, and team conversations. Knowing what they are, what they replace, and when to reach for them is itself a valuable skill.

C.1 The shape of a modern analytics stack



C.2 Faster local analytics: DuckDB, Polars, Arrow, Parquet

- **DuckDB** is an in-process SQL engine – think SQLite but column-oriented and aimed at analytical queries. Reads CSV, Parquet, and JSON directly. Often replaces a small Postgres database for local work. From R, use `duckdb` package; from Python, just `import duckdb`.
- **Polars** is a Pandas-like DataFrame library written in Rust. It is 5–100× faster than Pandas on most operations and uses a lazy execution model. The Python world has been quietly migrating to it.
- **Arrow** is the in-memory columnar format that Polars, DuckDB, and many modern engines share. Zero-copy data exchange between R, Python, and SQL engines.

- **Parquet** is the disk format. Columnar, compressed, self-describing. Smaller and faster than CSV by an order of magnitude. Read it from any of the above.

Pro Tip

A pragmatic upgrade path for a beginner: when your CSVs cross a few hundred megabytes, switch them to Parquet and query them with DuckDB. You will not need a database server until your data is several gigabytes.

C.3 Transformation: dbt and friends

dbt (data build tool) is the de-facto standard for managing SQL transformations in a warehouse. Instead of one analyst writing queries directly in a BI tool, the team writes SQL "models" versioned in git, tested for data quality, and built into a DAG of dependencies.

```sql
-- models/marts/customers_lifetime_value.sql
{{ config(materialized='table') }}

with orders as (
  select * from {{ ref('stg_orders') }}
),
agg as (
  select customer_id, sum(amount) as ltv
  from orders
  group by customer_id
)
select * from agg
```

Listing C.1: *A trivial dbt model*

The double braces (`{{ ref() }}`) are dbt's templating – a transform of one model can declare its inputs by name, and dbt constructs the build order automatically.
Closely related, less established: **SQLMesh** (newer, with better state management), **Coalesce** (visual builder for Snowflake).

C.4 Cloud warehouses

Warehouse	Strengths and where you see it
BigQuery	Google-native, serverless, pay-per-query. Common in India / Europe startups. SQL dialect close to standard.
Snowflake	Multi-cloud, fast for mid-to-large enterprises. Decoupled storage and compute.
Redshift	AWS-native; mature; older syntax. Common in AWS-only enterprises.
Databricks	Lakehouse blend of warehouse and Spark cluster. Strong with Python and large-scale ML.
Microsoft Fabric	Microsoft's unified analytics platform. Power BI, OneLake, and Synapse in a single product.

The SQL you learned in Chapters 2–3 transfers to all of them with small dialect tweaks. The skill that does not transfer for free is *cost-conscious querying* – in a pay-per-query world, an unbounded `SELECT *` can be the difference between an INR 10 and an INR 10,000 query.

C.5 Semantic layers

A **semantic layer** centralises business definitions ("revenue", "active user", "churn") so that every BI tool, notebook, and dashboard agrees on the same numbers. Without one, the marketing team's "churn rate" is rarely the same as the finance team's.

- **Cube** – open source, deployable anywhere.
- **dbt Semantic Layer** – built on top of dbt metrics.
- **LookML** – Looker's classic semantic layer; influential even outside Looker.
- **AtScale** – enterprise focus, BI-tool agnostic.

C.6 Python for analysts (a short note)

This workbook teaches R because R Shiny is a strong portfolio piece and statistics in R is more idiomatic. But many companies are Python-first. The good news: every dplyr pattern has a Pandas / Polars equivalent, and once you are fluent in one you will pick up the other in a couple of weekends. Aim to be *bilingual* within a year of starting your first job.

C.7 Workflow / orchestration: things that schedule everything

For when your pipeline becomes more than a single script:

- **Airflow** – the long-time standard. Python-defined DAGs.
- **Dagster** – newer, type-safe, asset-oriented.
- **Prefect** – friendlier developer experience.
- **cron + git + dbt Cloud** – the minimum viable stack for most small teams.

C.8 What to learn next, and in what order

If you finished the main book and want a clear next-90-days roadmap:

1. **Weeks 13–16:** Python and Pandas. You already know the concepts; the syntax is the only delta.
2. **Weeks 17–20:** A real cloud warehouse, hands-on. BigQuery's free tier is the easiest entry.
3. **Weeks 21–24:** dbt with that warehouse. Build a small project end-to-end.
4. **Weeks 25–28:** DuckDB + Polars for local fast analytics.
5. **Anytime:** contribute to one open-source dataset, blog post, or visualisation a month. This is what makes you findable.

The tools will keep changing. The skills underneath – asking the right question, picking the right method, communicating the answer clearly – will not.