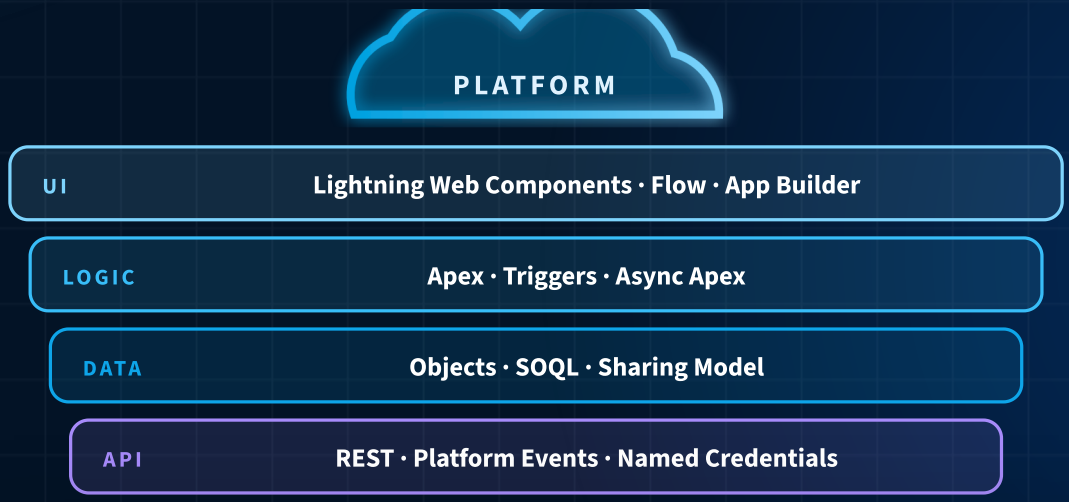
JOB-READY SERIES · JOB-SWITCH EDITION

ADMIN + DEVELOPER

Salesforce

The Job-Switch Guide · From Admin to Developer

Everything Salesforce interviews ask, in simple language: data model, security, Flow, reports, Apex, SOQL, triggers, governor limits, LWC, integration and DevOps, with org labs, real scenarios, projects and 120+ questions with answers.



CareerTrigger.trigger

```
trigger CareerTrigger on Candidate__c (after update) {
    for (Candidate__c c : Trigger.new) {    // bulkified, 0 SOQL in loop
        if (c.Skills__c == 'Admin + Dev') c.Status__c = 'Hired';
    }
}
```

Data Model · Security · Flow · Reports · Sales Cloud · Service Cloud · Apex · SOQL · Triggers · Async Apex · LWC · Integration · SF CLI · Agentforce

WRITTEN BY

Shreyash Tripathi

17 Chapters · Org Labs · Apex + LWC · Interview Questions

Edition 2026

JOB-READY SERIES · JOB-SWITCH EDITION



Salesforce

The Job-Switch Guide, from Admin to Developer

Learn the feature | Build it in your org | Write the Apex | Ship the project | Ace the interview

WRITTEN BY

Shreyash Tripathi

Job-Switch Edition, 2026

© Shreyash Tripathi. All rights reserved.

Preface

Salesforce runs the sales, service and customer operations of more than a hundred thousand companies, and the jobs around it (Admin, Developer, Consultant) are some of the most in-demand switches in the industry. The platform looks huge, but interviews keep returning to the same core: how data is modelled, who can see what, how to automate with Flow, when to write Apex, how to stay inside governor limits, how to build Lightning Web Components, and how to integrate and deploy safely.

This book teaches that core like a good school textbook: numbered chapters, simple language, a check with answers after every topic, and exercises with a full answer key. Because Salesforce is learned by building, every chapter has hands-on labs for a free Developer Edition org, Interview Corners, At Work stories from real projects, troubleshooting sections, a resume line, and the certification each topic maps to.

Build the three portfolio projects in Chapter 16, put your code on GitHub, and you will walk into interviews with real work to show instead of memorised definitions.

Shreyash Tripathi

Features of Every Chapter

🎯 **INTERVIEW CORNER**

Questions interviewers really ask, with model answers.

💼 **AT WORK**

What really happens on the job and on call.

🧪 **HANDS-ON LAB**

Do it yourself in a free Developer Edition org.

⚠️ **COMMON MISTAKE**

Traps that break production and fail interviews.

💡 **DO YOU KNOW?**

Real facts from the industry.

😄 **FUNNY SIDE**

Because "Too many SOQL queries: 101" needs jokes.

✅ **CHECK YOUR PROGRESS**

Short questions with answers after every topic, plus exercises, scenario questions, a mini project and a full answer key at the end of every chapter.

Contents

PART I: ADMIN FOUNDATIONS

1 The Salesforce Ecosystem and Career Roadmap 6

2 Platform Basics and the Data Model 24

3 Security and the Sharing Model 42

4 Automation with Flow and Declarative Tools 61

PART II: BUSINESS CLOUDS

5 Reports, Dashboards and Data Management 80

6 Sales Cloud and Service Cloud Essentials 96

PART III: DEVELOPER CORE

7 Apex Fundamentals 114

8 SOQL and SOSL 135

9 Apex Triggers and Governor Limits 154

10 Asynchronous Apex and Testing 175

PART IV: UI, INTEGRATION AND DELIVERY

11 Lightning Web Components I 199

12 Lightning Web Components II: Data and Apex 223

13 Integration 245

14 Salesforce DevOps and Deployment 264

15 Experience Cloud, Data Cloud and Agentforce 285

PART V: GET THE JOB

16 Salesforce Projects and Real Scenarios 304

17 Salesforce Interview Question Bank 326

APPENDICES

A Apex, SOQL and sf CLI Cheat Sheet 344

B Code Templates 347

C Glossary 354

D Job Switch Toolkit 358

PART I

Admin Foundations

The ecosystem, the data model, security and automation with Flow.

CHAPTER 1

The Salesforce Ecosystem and Career Roadmap

"Every company says it is customer-first. Salesforce is the software that checks whether they mean it."

After studying this chapter, you will be able to

- Explain what CRM and Salesforce are, in plain words, to a non-technical interviewer.
- Describe the multi-tenant, metadata-driven platform and why it leads to governor limits.
- Name the main Salesforce clouds and the business problem each one solves.
- Compare the Admin, Developer, Consultant, Business Analyst and Architect roles and what each interview tests.
- Plan a certification path and a 90-day switch plan with a portfolio and a resume that gets shortlisted.
- Set up a free Developer Edition org and the Salesforce CLI (sf).

Think of your neighbourhood kirana store uncle. He knows that Mrs. Sharma buys two litres of milk every morning, that the Iyer family wants filter coffee powder on the first of the month, and that the boy in flat 304 always forgets to pay on time. That memory in his head is a Customer Relationship Management system. It works beautifully for 200 customers. Now imagine a bank with 5 crore customers, 40,000 employees and 3,000 branches. Nobody's head is that big. That is the job Salesforce does.

This chapter is your map. Before you learn Apex, Flow or LWC, you need to know what the platform is, who uses it, which roles exist, which certificates matter and how to go from "I have heard of Salesforce" to "I have an offer letter" in about three months. Keep a notebook open. By the end, you will have your own written plan.

1.1 What Are CRM and Salesforce?

CRM stands for **Customer Relationship Management**. A CRM system stores everything a company knows about its customers: who they are, what they bought, what they complained about, which salesperson is talking to them and what was promised. It turns "Ravi in sales remembers" into "the system remembers".

Salesforce is the largest CRM company in the world. It was founded in 1999 in San Francisco by Marc Benioff and a few co-founders. Its big idea was simple: instead of installing CRM software on

your own servers, you open a browser and log in. No CDs, no servers, no upgrades done on a Sunday night. This model is called **Software as a Service (SaaS)**.

1.1.1 The core business objects

Almost every Salesforce interview starts with the sales process. Learn these standard objects by heart:

Object	What it stores	Everyday example
Lead	A possible customer, not yet qualified	A parent who filled an enquiry form on a coaching website
Account	A company or household you do business with	Infosys, or "The Sharma Family"
Contact	A person linked to an account	Priya Sharma, the HR head at that account
Opportunity	A potential deal with an amount and stage	"200 laptops, Rs 1.2 crore, Negotiation stage"
Case	A customer problem or support ticket	"My order arrived broken"
Campaign	A marketing effort	"Diwali sale email blast"

When a lead is qualified, you **convert** it. Conversion creates an Account, a Contact and (optionally) an Opportunity in one click. This "lead to cash" story is the spine of Sales Cloud.

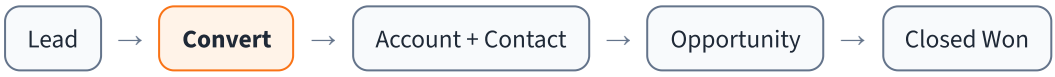


Figure 1.1: The classic lead-to-cash journey in Sales Cloud

> 💡 **DO YOU KNOW?**
>
> Salesforce has led the worldwide CRM market by revenue share for more than ten years in a row, according to IDC reports. Its yearly revenue is now in the range of about USD 35 billion or more. The annual Dreamforce conference in San Francisco draws over 40,000 people in person.

> 🎯 **INTERVIEW CORNER**
>
> **Q:** Explain Salesforce to my grandmother. **A:** It is a giant, shared online register where a company writes down every customer, every sale and every complaint, so that any employee can see the full story and serve the customer better.
>
> **Q:** What happens when you convert a lead? **A:** Salesforce creates an Account, a Contact and optionally an Opportunity, maps the lead fields to them using lead field mapping, and marks the lead as converted so it no longer shows in normal lead views.

CHECK YOUR PROGRESS 1.1

1. **What does SaaS mean in the context of Salesforce?**
   Answer: The software runs on Salesforce's servers and you use it through a browser or mobile app on a subscription, with no installation or server maintenance.
2. **Which object stores a support ticket?**
   Answer: The Case object.
3. **Name the three records lead conversion can create.**
   Answer: Account, Contact and Opportunity (the opportunity is optional).

1.2 The Platform: Multi-Tenant and Metadata-Driven

To pass a developer interview, you must explain *why* Salesforce behaves the way it does. Two words explain almost everything: multi-tenant and metadata-driven.

1.2.1 Multi-tenancy

Think of an apartment building. Every family has its own locked flat, but they share the lift, the water tank, the generator and the security guard. In Salesforce, each customer company gets its own **org** (short for organisation), which is like a flat. Thousands of orgs share the same servers, database and application code underneath. That is **multi-tenancy**.

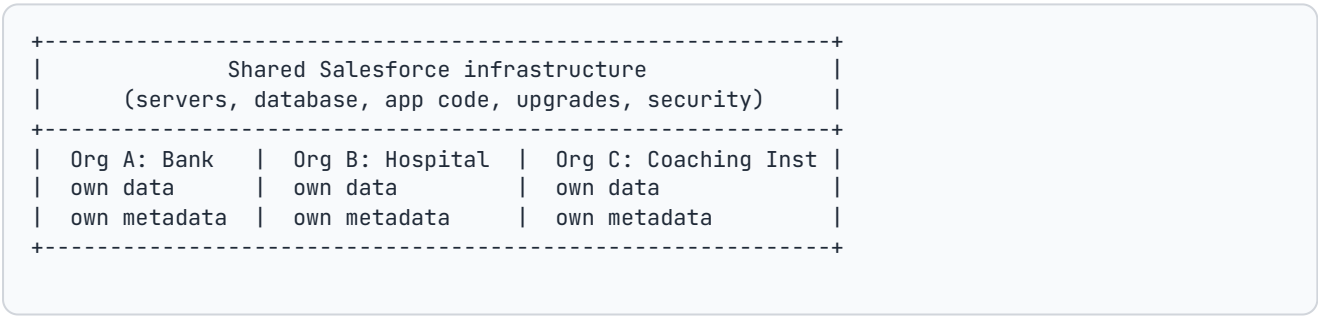


Figure 1.2: Many tenants, one building

Because the lift is shared, the society makes rules: no carrying a sofa in the lift during office hours. Salesforce does the same with **governor limits**. For example, in a single synchronous Apex transaction you can run at most **100 SOQL queries** and **150 DML statements**. Asynchronous code gets a higher SOQL limit (200). These limits stop one badly written org from slowing everyone else down. You will meet them again and again in later chapters.

1.2.2 Metadata-driven

When you create a custom object called `Student__c`, Salesforce does not create a new physical table in an Oracle database just for you. Instead, it stores a **description** of your object (its name, fields, rules, layouts) as **metadata**. At runtime, the platform reads that metadata and builds your app on the fly. Your customisations live separately from Salesforce's core code, which is why Salesforce can upgrade every org three times a year without breaking your app.

Metadata matters on the job because everything you build (objects, fields, flows, Apex classes, page layouts) is metadata that can be retrieved, stored in Git and deployed to another org with the Salesforce CLI.

```
$ sf project retrieve start --metadata CustomObject:Student__c --target-org dev
Retrieving v62.0 metadata from admin@dev-org.com using the v62.0 SOAP API
Preparing retrieve request... Succeeded

 Retrieved Source
 State    Name        Type          Path
 Created  Student__c  CustomObject  force-
app/main/default/objects/Student__c/Student__c.object-meta.xml
```

(The API version in the output depends on your CLI and org. Do not worry if yours shows a different number.)

🔑 **KEY POINT**

Multi-tenant means shared resources, which means governor limits. Metadata-driven means your app is a description, which means it can be moved between orgs and upgraded safely. If you can say these two lines confidently, you will already sound better than half the candidates.

🎯 **INTERVIEW CORNER**

Q: Why does Salesforce have governor limits? **A:** Because it is multi-tenant. Many orgs share the same resources, so the platform caps what one transaction can consume, such as 100 SOQL queries and 150 DML statements per synchronous transaction.

Q: What is metadata in Salesforce? **A:** Data that describes your configuration: object and field definitions, layouts, flows, Apex code, profiles. It can be retrieved and deployed with the Metadata API, usually through the sf CLI.

✅ **CHECK YOUR PROGRESS 1.2**

1. **Give a real-life analogy for multi-tenancy.**
   Answer: An apartment building: separate locked flats (orgs) but a shared lift, water and security (infrastructure).
2. **What is the SOQL query limit in a synchronous Apex transaction?**
   Answer: 100 SOQL queries (200 in asynchronous Apex).

1.3 The Main Salesforce Clouds

Salesforce sells many products, called "clouds". They all sit on the same core platform (sometimes called the Customer 360 platform), so a skill you learn on one cloud usually helps on another.

Cloud	Problem it solves	Key features
Sales Cloud	Track leads and deals, forecast revenue	Leads, Opportunities, Forecasts, Quotes, Territory Management
Service Cloud	Customer support at scale	Cases, Omni-Channel routing, Knowledge, Entitlements, Service Console
Marketing Cloud	Email, SMS and journey marketing	Journey Builder, Email Studio, Account Engagement (formerly Pardot)
Commerce Cloud	Online stores (B2C and B2B)	Storefronts, carts, product catalogues, order management
Experience Cloud	Portals and websites for customers or partners	Customer portals, partner portals, help centres (formerly Community Cloud)
Data Cloud	Unify customer data from many systems in near real time	Data streams, identity resolution, segments, calculated insights. Recently rebranded as Data 360; check the docs for the current name
Agentforce	AI agents that act inside Salesforce	Agents with topics and actions, grounded on CRM data, built in Agent Builder

1.3.1 Where should a beginner focus?

Most jobs in India and abroad are on **Sales Cloud and Service Cloud** plus the core platform (objects, security, Flow, Apex, LWC). Start there. Marketing Cloud and Commerce Cloud are separate skill sets with their own tools and are better as a second specialisation. Agentforce and Data Cloud are growing fast; an Agentforce certification on top of a solid core makes your profile stand out.

> 😄 **FUNNY SIDE**
>
> Salesforce renames products so often that senior consultants keep a translation table in their heads: "Community Cloud? You mean Experience Cloud. Pardot? You mean Account Engagement. Einstein Copilot? You mean Agentforce." If an interviewer uses an old name, smile and use the new one in your answer. It shows you are current.

> ✅ **CHECK YOUR PROGRESS 1.3**
>
> 1. **Which cloud would a company use to let customers log in and track their own support cases?**
>   **Answer:** Experience Cloud (a customer portal), usually together with Service Cloud.
> 2. **What was Pardot renamed to?**
>   **Answer:** Marketing Cloud Account Engagement.
> 3. **Which clouds should a beginner focus on first?**
>   **Answer:** Sales Cloud and Service Cloud on the core platform.

1.4 Roles in the Ecosystem and What Each Interviews For

"Salesforce job" is not one job. Pick a target role before you study, because the interview for each one is different.

Role	Daily work	Interview focuses on	Key certification
Administrator	Users, security, fields, reports, dashboards, Flow, data loads	Sharing model, profiles vs permission sets, validation rules, Flow, reports	Administrator
Developer	Apex, triggers, SOQL, LWC, integrations, deployments	Governor limits, bulkified triggers, test classes, async Apex, LWC lifecycle, live coding	Platform Developer I, then II; JavaScript Developer
Consultant	Gather requirements, design solutions, configure, run workshops	Cloud features, solution design, client handling scenarios	Sales Cloud or Service Cloud Consultant
Business Analyst	User stories, process mapping, UAT, documentation	Requirement gathering, process maps, acceptance criteria	Business Analyst
Architect	Data model, security, integration and scaling design	Large data volumes, sharing at scale, integration patterns, trade-offs	Application / System Architect, CTA

1.4.1 Which role fits a developer switching jobs?

If you already code in Java, C#, JavaScript or Python, the **Salesforce Developer** path is the natural switch. Apex looks a lot like Java. LWC is standard JavaScript with web components. Your existing skills in Git, testing and APIs carry over directly. But do not skip admin skills: good Salesforce developers know when *not* to write code. Many interviewers ask "could this be done with Flow?" before they ask about Apex.

1.4.2 What a typical developer interview checks

- Declarative vs programmatic: when Flow, when Apex.
- Writing a bulkified trigger with a handler class, no SOQL or DML inside loops.
- SOQL relationships (parent-to-child subqueries, child-to-parent dot notation).
- Test classes, 75% code coverage for deployment to production, `Test.startTest()` and `Test.stopTest()`.
- Async Apex: future, Queueable, Batch, Schedulable.
- LWC: decorators (`@api` , `@wire` , `@track`), events, calling Apex.
- Security in code: `with sharing` , user mode queries.

🎯 INTERVIEW CORNER

Q: You are a Java developer. Why should we hire you for a Salesforce developer role? **A:** Apex is Java-like, so I write clean object-oriented code from day one. I also learned the platform rules: governor limits, bulkification, the sharing model and when to use Flow instead of code. Here is a project on my GitHub that shows a trigger framework with tests at 90% coverage.

Q: What does an admin do that a developer does not? **A:** An admin manages users, security, data quality, reports and declarative automation day to day. A developer builds what configuration cannot, such as complex logic, integrations and custom UI. In small teams the same person often does both.

✅ CHECK YOUR PROGRESS 1.4

1. **Which certification usually comes first for a developer?**
   Answer: Platform Developer I (many also take Administrator first or alongside it).
2. **What minimum Apex code coverage is required to deploy to production?**
   Answer: 75% overall across the org, and every trigger needs at least some coverage.

1.5 Certifications, Trailhead and Your Developer Edition Org

1.5.1 The certification path

Salesforce certifications are proctored exams, taken online or at a test centre. They are respected by recruiters because they are hard to fake. A suggested order:



Figure 1.3: A developer-focused certification path

Certification	Tests	Covered in this book
Administrator	Setup, data model, security, Flow, reports, service and sales basics	Chapters 2, 3, 4
Platform App Builder	Data model, UI, declarative automation, app deployment	Chapters 2, 4
Platform Developer I	Apex, SOQL, triggers, testing, LWC basics, order of execution	Later developer chapters, plus Chapter 4 order of execution
JavaScript Developer	Modern JavaScript plus an LWC Trailhead superbadge requirement	LWC chapters
Platform Developer II	Advanced Apex, async, integration, performance, testing	Advanced chapters

Most associate and professional exams have about 60 scored questions and a time limit of around 105 minutes. Exam fees are roughly USD 200 for the main exams and about USD 100 for a retake, with lower prices for associate-level exams. These details change, so **check the current exam guide on Trailhead Academy** before booking. Salesforce also requires you to complete free **maintenance modules** on Trailhead once a year to keep each certification active.

1.5.2 Trailhead

Trailhead (trailhead.salesforce.com) is Salesforce's free learning site. It has modules (short lessons with a quiz or a hands-on challenge), trails (collections of modules), projects and **superbadges** (larger, scenario-based challenges). A Trailhead hands-on challenge is checked automatically against a real org. Ranks go from Scout up to Ranger and beyond. Recruiters do look at Trailhead profiles, but a few superbadges impress more than 300 easy badges.

Each Trailhead account can create **Trailhead Playgrounds**, which are free orgs linked to your profile for completing challenges.

1.5.3 Developer Edition org

A **Developer Edition** org is a free, full-featured Salesforce org that never expires as long as you log in occasionally (Salesforce may deactivate long-unused orgs, so log in now and then). Sign up at developer.salesforce.com. It has limited storage and only a couple of user licences, but it has Apex, Flow, LWC and most features you need to learn.

> 🧪 **HANDS-ON LAB 1.1**
>
> 1. Go to developer.salesforce.com and sign up for a free Developer Edition org. Use a username that looks like an email but need not be real, such as `yourname@dev-2026.com`. Your real email receives the activation link.
> 2. Log in, click the gear icon, then **Setup**. Search "Company Information" in Quick Find. Note your org ID and edition.
> 3. Install Node.js (LTS), then install the Salesforce CLI: `npm install --global @salesforce/cli`. Run `sf --version`.
> 4. Log in from the terminal: `sf org login web --alias dev --set-default`. A browser window opens; approve access.
> 5. Run `sf org display` and `sf org list`. Take a screenshot for your notes.
> 6. Install Visual Studio Code and the "Salesforce Extension Pack" extension.
> 7. Create a project: `sf project generate --name sf-journey`, open it in VS Code and push it to a new GitHub repository. This repo will hold every lab in this book.

```
$ sf org login web --alias dev --set-default
Successfully authorized yourname@dev-2026.com with org ID 00D5j00000XXXXXX

$ sf org list
     Alias  Username               Org Id            Connected Status
 __ _____ ____________________ __________________ _________________
(D) dev     yourname@dev-2026.com    00D5j00000XXXXXX   Connected
```

(The exact table style varies by CLI version. A marker such as a leaf icon or (D) shows your default org.)

> ⚠️ **COMMON MISTAKE**
>
> Beginners try to use the older `sfdx force:...` commands from old blog posts. The current CLI is `sf`, with commands like `sf project deploy start` and `sf org login web`. The old `sfdx` style is deprecated. If a tutorial shows `sfdx force:source:push`, look up the `sf` equivalent instead.

> ✅ **CHECK YOUR PROGRESS 1.5**
>
> 1. **What is a superbadge?**
>
>   **Answer:** A larger, scenario-based Trailhead challenge that tests applied skills in an org, more respected than a simple module badge.
>
> 2. **What must you do every year to keep a certification active?**
>
>   **Answer:** Complete the free maintenance module for that certification on Trailhead.
>
> 3. **Which command authorises an org in the sf CLI using the browser?**
>
>   **Answer:** `sf org login web`, for example with `--alias dev --set-default`.

1.6 Three Releases a Year

Salesforce ships three major releases every year: **Spring**, **Summer** and **Winter**. They are named with the next year's number for Winter, so the release that arrives around September or October 2025 is called **Winter '26**. Each release raises the API version by one (for example, API 64.0, 65.0 and so on).

1.6.1 How a release reaches your org

1. **Release notes** are published several weeks before the release. They are long; skim the sections for your clouds.
2. **Sandbox preview**: sandboxes on preview instances get the release about a month early, so you can test.
3. **Production upgrade**: orgs are upgraded over a few weekends. You do nothing; the platform upgrades itself.

> 💼 **AT WORK**
>
> Monday after a release weekend, the support team says the "Submit for Approval" button on Quotes is missing. The admin checks: the button was a JavaScript custom button, a Classic-era feature, and a Release Update the team had ignored was auto-enforced. Fix: replace the button with a Quick Action that launches a screen flow, test in a sandbox, deploy the same afternoon. Lesson: put "review Release Updates" on the team calendar before every release.

✅ **CHECK YOUR PROGRESS 1.6**

1. **Name the three Salesforce releases.**
   Answer: Spring, Summer and Winter.
2. **Why is a sandbox preview useful?**
   Answer: It lets you test your customisations on the new release about a month before production is upgraded.

1.7 The 90-Day Switch Plan

Ninety days is tight but realistic if you can give about 2 hours on weekdays and 5 to 6 hours on weekends. Adjust if you have less time; consistency beats speed.

Weeks	Focus	Output
1-2	Platform basics, data model, Trailhead Admin Beginner trail	Developer org, custom app with 3 related objects
3-4	Security, sharing, Flow, validation rules, reports	Three working flows, one dashboard
5	Admin exam revision, practice tests	Administrator certification booked or passed
6-7	Apex, SOQL, triggers, test classes	Trigger with handler and test class at 85%+ coverage
8-9	Async Apex, integration (REST callouts), LWC basics	One LWC calling Apex, one Queueable callout
10	PD1 revision, practice questions	Platform Developer I booked or passed
11	Portfolio polish, GitHub, resume, LinkedIn	Public repo with README and screenshots
12-13	Mock interviews, applications, referrals	20-30 targeted applications, 3-5 interviews

🔑 **KEY POINT**

Do not wait to "finish learning" before applying. Start applying around week 8 with Admin certification and a growing portfolio. Interviews themselves teach you what to study next.

1.7.1 Portfolio ideas

A portfolio project should solve a believable business problem, include both declarative and code pieces, and live in a public GitHub repo deployable with the sf CLI.

- **Coaching Institute CRM**: Students, Batches, Enrolments (junction object), fee reminders by scheduled flow, attendance LWC. (We design its data model in Chapter 2.)
- **Clinic Appointment Manager**: Patients, Doctors, Appointments, double-booking prevention with a trigger, an Experience Cloud page for patients.

- **Housing Society Helpdesk**: Cases for complaints, Omni-Channel style queue assignment, escalation flow, SLA dashboard.
- **Integration demo**: Apex REST callout to a public weather or currency API, stored in a custom object, with mock tests using `HttpCalloutMock`.

Each repo should have a README with: the business problem, a data model diagram, screenshots, a "how to deploy" section, and test coverage numbers.

```
## How to deploy
sf org login web --alias myorg
sf project deploy start --source-dir force-app --target-org myorg
sf org assign permset --name Coaching_Admin --target-org myorg
sf apex run test --test-level RunLocalTests --code-coverage --result-format human --target-
org myorg
```

1.7.2 Resume tips

- Put a **headline** at the top: "Salesforce Developer | Platform Developer I | Apex, LWC, Flow".
- List certifications near the top, with month and year. Add your Trailhead profile link.
- Rewrite past experience in transferable terms. "Built Java REST services" becomes relevant for integration roles.
- Use numbers: records migrated, hours saved, coverage percentage, users supported.
- Name real features: "record-triggered flow", "permission set groups", "Queueable Apex", not "worked on automation".
- Keep it to one or two pages. Recruiters scan for keywords first.

1.7.3 Typical interview rounds

1. **Recruiter screen** (15–30 minutes): notice period, salary, certifications, basic fit.
2. **Technical round 1**: concepts. Data model, sharing, Flow vs Apex, governor limits, trigger best practices.
3. **Technical round 2**: hands-on. Write a trigger or SOQL on a shared screen, or explain an LWC. Some companies give a take-home task in a Developer org.
4. **Scenario or design round**: "A sales manager cannot see her team's opportunities. Why?" or "Design a solution for X".
5. **Managerial / HR round**: teamwork, handling production issues, career goals.

🎯 **INTERVIEW CORNER**

Q: Tell me about a Salesforce project you built. **A:** Use the STAR format: Situation (coaching institute losing track of fee dues), Task (automate reminders and show dues per batch), Action (custom objects, master-detail, roll-up summary, scheduled flow, an Apex trigger for duplicate enrolment check with tests), Result (deployable repo, 92% coverage, reminders sent automatically).

Q: Why do you want to move to Salesforce? **A:** Give a real reason: strong demand, a clear certification path, and you enjoy building business solutions quickly. Then show proof: certificates and projects.

✅ **CHECK YOUR PROGRESS 1.7**

1. **What four things should a portfolio README contain?**
   Answer: The business problem, a data model diagram, screenshots, deployment steps (and test coverage numbers).
2. **When should you start applying in the 90-day plan?**
   Answer: Around week 8, not after "finishing" everything.
3. **Which interview format is recommended for project questions?**
   Answer: STAR: Situation, Task, Action, Result.

1.8 Troubleshooting Your Setup

Your first week will have small setup problems. Here are the common ones.

Symptom	Cause	Fix
`sf: command not found`	CLI not installed or not on PATH	Reinstall with `npm install --global @salesforce/cli`; restart the terminal; on Windows check that the npm global folder is on PATH
Browser login opens but the CLI hangs	Local callback port (1717) blocked or in use	Close other CLI sessions, allow the port in firewall, or use `sf org login web` again; for headless machines use JWT or device login, check the docs for your version
`No default environment found`	No default target org set	`sf config set target-org dev` or pass `--target-org dev`
`INVALID_LOGIN` on login	Wrong username, or using production login URL for a sandbox	Use `--instance-url https://test.salesforce.com` for sandboxes, or your My Domain URL

💼 **AT WORK**

A new joiner cannot deploy and keeps getting `INVALID_SESSION_ID`. The team lead runs `sf org list` and sees the org status as "Expired" because the access token was revoked when the user's password was reset. Fix: `sf org logout --target-org uat`, then `sf org login web --alias uat --instance-url https://test.salesforce.com`. Ten minutes, problem gone.

✅ **CHECK YOUR PROGRESS 1.8**

1. **Which login URL do you use for a sandbox?**
   Answer: https://test.salesforce.com (or the org's My Domain URL).
2. **How do you set a default org for the CLI?**
   Answer: `sf config set target-org <alias>`, or use `--set-default` when logging in.

What Have We Learnt?

- CRM stores everything about customers; Salesforce is the market-leading cloud CRM, founded in 1999.
- Core sales objects are Lead, Account, Contact, Opportunity, Case and Campaign; lead conversion creates Account, Contact and optionally Opportunity.
- Salesforce is multi-tenant, so it enforces governor limits such as 100 SOQL queries and 150 DML statements per synchronous transaction.
- Salesforce is metadata-driven; your customisations are metadata that can be retrieved and deployed with the sf CLI.
- Main clouds: Sales, Service, Marketing, Commerce, Experience, Data Cloud and Agentforce, all on one core platform.
- Roles include Admin, Developer, Consultant, Business Analyst and Architect, each with a different interview focus.
- A developer path: Administrator, Platform App Builder, Platform Developer I, JavaScript Developer, Platform Developer II.
- Trailhead and a free Developer Edition org are your practice ground; certifications need yearly maintenance modules.
- Salesforce has three releases a year (Spring, Summer, Winter); admins test in preview sandboxes and review Release Updates.
- A focused 90-day plan, a GitHub portfolio and a keyword-rich resume with numbers get you to interviews.

Key terms: CRM, SaaS, org, multi-tenancy, metadata, governor limits, Sales Cloud, Service Cloud, Experience Cloud, Data Cloud, Agentforce, Trailhead, superbadge, Developer Edition, sandbox preview, Release Update, sf CLI, STAR

format.

🔑 **RESUME LINE**

"Built a Coaching Institute CRM on Salesforce (5 custom objects, 3 record-triggered flows, 1 Apex trigger with 92% test coverage), deployable via sf CLI from a public GitHub repo."
"Earned Salesforce Administrator and Platform Developer I certifications within 90 days while working full time as a Java developer."

Exercises

A. Multiple Choice Questions

1. Why does Salesforce enforce governor limits?
   a. To sell more licences
   b. Because resources are shared in a multi-tenant architecture
   c. Because Apex is slow
   d. Only for free orgs
2. Lead conversion can create which records?
   a. Case and Contact
   b. Account, Contact and optionally Opportunity
   c. Only an Opportunity
   d. Campaign and Account
3. Which cloud is used to build customer and partner portals?
   a. Marketing Cloud
   b. Commerce Cloud
   c. Experience Cloud
   d. Data Cloud
4. How many SOQL queries are allowed in a synchronous Apex transaction?
   a. 50
   b. 100
   c. 150
   d. 200
5. Which is NOT one of the three yearly releases?
   a. Spring
   b. Summer
   c. Autumn
   d. Winter
6. Which command creates a new Salesforce DX project?
   a. `sf project generate --name myproj`
   b. `sfdx new project`
   c. `sf org create project`
   d. `npm init salesforce`

B. Short Answer Questions

1. Explain "metadata-driven" and why it helps upgrades.
2. Compare the Admin and Developer roles in three points each.
3. What is the difference between a Trailhead Playground and a Developer Edition org?
4. List five things that make a Salesforce resume stronger.

C. Scenario Questions

1. The morning after a release weekend, users report that a custom button on the Opportunity page no longer works. What do you do?
2. A new teammate cannot log in to a sandbox from the CLI and gets `INVALID_LOGIN`. Production login works. How do you help?

3. Your manager asks why a simple data fix script failed with "Too many SOQL queries: 101" when it ran on 500 records. Explain and suggest a fix.

D. Mini Project

Create a public GitHub repo called `sf-journey`. Generate a Salesforce DX project inside it, connect your Developer Edition org, and write a README with: your target role, your 90-day plan as a table, your certification target dates, and the commands you used. Add one custom object (any business idea) and retrieve its metadata into the repo with the sf CLI.

Answer Key: Chapter 1

A. MCQ

1. (b) Shared resources in a multi-tenant system need fair-use caps.
2. (b) Account, Contact and optionally Opportunity.
3. (c) Experience Cloud builds portals and sites.
4. (b) 100 synchronous (200 asynchronous).
5. (c) There is no Autumn release; Winter covers that period.
6. (a) `sf project generate` is the current command.

B. Short Answers

1. Your org's customisations are stored as descriptions (metadata) separate from Salesforce's core code. The platform builds the app at runtime from metadata, so Salesforce can upgrade the core for everyone without overwriting your configuration.
2. Admin: manages users and security, builds fields, flows and reports, keeps data clean. Developer: writes Apex and LWC, builds integrations, handles deployments and test coverage. Both must understand the data model and sharing.
3. A Trailhead Playground is a free org linked to your Trailhead account for completing challenges. A Developer Edition org is a free standalone org from developer.salesforce.com, often used for personal projects and portfolio work. Both are similar in features.
4. Certifications at the top with dates; a clear headline; quantified achievements; named features (record-triggered flow, Queueable Apex); links to Trailhead and GitHub projects.

C. Scenario Questions

1. (1) Reproduce the issue and note the error. (2) Check the release notes and Setup, Release Updates for anything enforced this weekend. (3) Inspect the button: if it is a JavaScript button, it will not work in Lightning. (4) Build a replacement (Quick Action with a screen flow or LWC) in a sandbox and test. (5) Deploy, inform users, and add release-update reviews to the release checklist.
2. (1) Confirm the username (sandbox usernames usually have a suffix like `.uat`). (2) Use the sandbox login URL: `sf org login web --alias uat --instance-url https://test.salesforce.com`, or the sandbox My Domain URL. (3) If the sandbox was refreshed, the password may be the production one at refresh time; reset if needed.
3. The script ran a SOQL query inside a loop, so 500 records meant 500 queries, crossing the 100 limit. Fix: collect IDs into a Set first, run one query with `WHERE Id IN :ids`, store results in a Map and loop over the map.

D. Mini Project: Model Solution

Repo structure: `sf-journey/README.md`, `sfdx-project.json`, `force-app/main/default/objects/Idea__c/Idea__c.object-meta.xml` and its `fields/` folder. Commands: `sf project generate --name sf-journey`, `sf org login web --alias dev --set-default`, create the object in Setup, then `sf project retrieve start --metadata CustomObject:Idea__c`. README

sections: Target Role, 90-Day Plan table, Certification Dates, Setup Commands, Screenshots. Commit and push with a clear message such as "Add Idea__c object metadata".

CHAPTER 2

Platform Basics and the Data Model

"A bad data model is like a badly planned house. You can paint it, but you cannot fix the plumbing."

After studying this chapter, you will be able to

- Explain orgs, editions and sandbox types, and pick the right sandbox for a task.
- Create custom objects and fields with the correct data types and API names.
- Choose between lookup, master-detail, many-to-many and hierarchical relationships.
- Build roll-up summary and formula fields, record types, page layouts and Lightning record pages.
- Design a complete data model for a real business, step by step, and query it with SOQL.

Think of a school office. There is a register of students, a register of classes, a fee register and an attendance register. Each register has columns, and some columns point to other registers ("Roll No. 42 is in Class 9B"). If the clerk designs these registers badly, every report at the end of the year becomes a nightmare. In Salesforce, registers are **objects**, columns are **fields** and the pointers are **relationships**.

Every other skill in this book (security, Flow, Apex, LWC) sits on top of the data model. Interviewers know this, so data model questions appear in the Administrator, Platform App Builder and Platform Developer I exams, and in almost every interview. Get this chapter right and the rest becomes easier.

2.1 Orgs, Editions and Sandboxes

An **org** is one instance of Salesforce for one company: its own data, users, metadata and URL (for example `https://acme.my.salesforce.com`). A company usually has one **production** org, where real users work, and several **sandboxes**, which are copies used for building and testing.

2.1.1 Editions

The **edition** decides which features and limits you get. Common ones are Professional, Enterprise and Unlimited, plus small-business bundles (Starter and Pro Suite). Enterprise is the most common in companies; it includes API access, full Flow and Apex. Professional edition limits API access and code unless bought as add-ons. Developer Edition is the free one for learning. Exact packaging changes often, so check the pricing page for your situation.

2.1.2 Sandbox types

Type	Copies	Data storage	Refresh interval	Use it for
Developer	Metadata only	200 MB	1 day	Individual development
Developer Pro	Metadata only	1 GB	1 day	Development with more test data
Partial Copy	Metadata + sample data (via template)	5 GB	5 days	QA, integration testing
Full	Metadata + all data	Same as production	29 days	UAT, performance tests, staging

Developers also use **scratch orgs**: short-lived orgs created from a config file with the sf CLI, perfect for source-driven development.

```
$ sf org create scratch --definition-file config/project-scratch-def.json --alias feature1 --
duration-days 7 --target-dev-hub devhub
Creating Scratch Org... done
Your scratch org is ready.
```

> 🎯 **INTERVIEW CORNER**
>
> **Q:** Which sandbox would you use for user acceptance testing? **A:** A Full sandbox, because it has all production data and metadata, so users test with realistic records. If budget does not allow, a Partial Copy with a good data template.
>
> **Q:** What is the difference between a sandbox and a scratch org? **A:** A sandbox is a copy of production that lives until you refresh or delete it. A scratch org is a temporary, empty org built from source and a definition file, usually for one feature, and it expires (up to 30 days).

> ✅ **CHECK YOUR PROGRESS 2.1**
>
> 1. **Which sandbox copies all production data?**
>
>   **Answer:** The Full sandbox.
>
> 2. **How often can a Developer sandbox be refreshed?**
>
>   **Answer:** Once a day.

2.2 Standard and Custom Objects

Standard objects come with Salesforce: Account, Contact, Lead, Opportunity, Case, User, Task, Event and many more. **Custom objects** are ones you create for your business, like `Student__c` or `Batch__c`.

2.2.1 API names

Every object and field has a **label** (what users see) and an **API name** (what code uses). Custom objects and custom fields end in `__c`. Relationship names used in SOQL end in `__r`. Labels can change freely; API names are referenced by code, flows and integrations, so changing them later is painful.

Thing	Label	API name
Standard object	Account	`Account`
Custom object	Student	`Student__c`
Custom field	Date of Birth	`Date_of_Birth__c`
Relationship (in SOQL)	Batch	`Batch__r`

Every record gets a unique **record ID**: 15 characters (case-sensitive) or 18 characters (case-insensitive, safe for Excel). The first three characters are the key prefix: `001` for Account, `003` for Contact, `006` for Opportunity, `005` for User.

Every object also gets standard fields automatically: `Id`, `Name`, `OwnerId` (unless it is a detail in master-detail), `CreatedById`, `CreatedDate`, `LastModifiedById`, `LastModifiedDate` and `SystemModstamp`.

```
$ sf sobject describe --sobject Student__c --target-org dev | grep '"name"' | head -5
    "name": "Id",
    "name": "OwnerId",
    "name": "IsDeleted",
    "name": "Name",
    "name": "CreatedDate",
```

> 💡 **DO YOU KNOW?**
>
> The 18-character ID exists because Excel and many tools treat "a01ABC" and "a01abc" as the same text. The extra three characters encode the capitalisation of the first 15, so the 18-character version is safe in case-insensitive tools. In formulas, `CASESAFEID(Id)` returns it.

✅ **CHECK YOUR PROGRESS 2.2**

1. **What suffix do custom fields have?**
   Answer: `__c`.
2. **What is the key prefix of an Opportunity ID?**
   Answer: `006`.

2.3 Fields and Data Types

Picking the right data type matters because changing it later can lose data, break formulas and fail deployments.

Data type	Stores	Notes
Text	Up to 255 characters	Names, codes
Text Area (Long)	Up to 131,072 characters	Not usable in SOQL WHERE filters
Rich Text Area	Formatted text and images	Notes, descriptions
Number, Currency, Percent	Numeric values	Set length and decimal places
Date, Date/Time, Time	Dates and times	Date/Time is stored in UTC, shown in user's time zone
Checkbox	True or false	Never null; defaults to unchecked
Picklist	One value from a list	Use restricted picklists to block bad values
Multi-Select Picklist	Several values	Hard to report and filter on; avoid when possible
Email, Phone, URL	Contact info	Email is format-validated; clickable in UI
Auto Number	System-generated sequence	Example format: `STU-{0000}`
Formula	Calculated, read-only	Computed when read, not stored
Roll-Up Summary	COUNT, SUM, MIN, MAX of children	Master-detail parents only
Lookup, Master-Detail	Link to another record	See section 2.4
Geolocation	Latitude and longitude	Used for distance formulas

2.3.1 Field properties

Fields can be **Required**, **Unique**, **External ID** (used for upserts from other systems), and have a **default value** and **help text**. Mark an ID from an outside system, like an ERP customer code, as External ID and Unique. Data loads can then `upsert` on it without creating duplicates.

```
$ sf data upsert bulk --sobject Student__c --file students.csv --external-id Admission_No__c
--target-org dev --wait 10
Status: JobComplete
Processed records: 1200 | Successful: 1198 | Failed: 2
```

> ⚠ **COMMON MISTAKE**
>
> Storing phone numbers in a Number field. The leading zero disappears, "+91" fails, and someone will add a phone number to another phone number one day. Use the Phone type (or Text) for anything you do not do maths on: phone numbers, PIN codes, Aadhaar-style IDs.

CHECK YOUR PROGRESS 2.3

1. **Why mark a field as External ID?**
   Answer: So data loads and integrations can upsert using an ID from another system, matching existing records instead of creating duplicates.
2. **Why avoid multi-select picklists?**
   Answer: They are hard to report, filter and use in formulas; a related object or separate checkboxes are often better.

2.4 Relationships

Relationships connect objects. A relationship field always sits on the **child** (the "many" side) and points to the parent.

2.4.1 Lookup relationship

A loose link. The child has its own owner and sharing. The lookup can be optional. When the parent is deleted, you choose: clear the field (default), or block deletion. Example: Case looks up to Account.

2.4.2 Master-detail relationship

A tight link, like a bill and its line items. The detail cannot exist without its master:

- The master field is always required.
- Deleting the master deletes all details (cascade delete).
- The detail has no owner; its sharing and security come from the master.
- The master can have **roll-up summary** fields over the details.
- An object can have at most **two** master-detail relationships.
- Reparenting (changing the master) is off by default; you can allow it in the field settings.

2.4.3 Many-to-many with a junction object

A student can join many batches; a batch has many students. Salesforce has no direct many-to-many field. You create a **junction object** (for example `Enrolment__c`) with **two master-detail fields**, one to each parent. The first master-detail you create is the primary one; it controls the junction record's look and feel, and the junction's sharing depends on access to both parents.

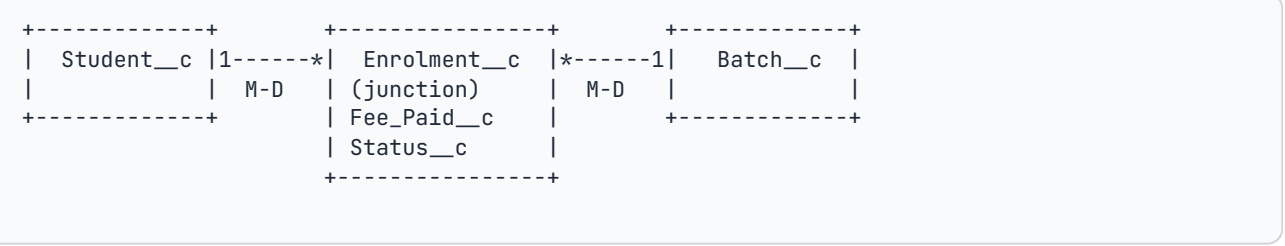


Figure 2.1: Junction object linking Students and Batches

2.4.4 Hierarchical relationship

A special lookup available **only on the User object**, pointing from a user to another user. The standard Manager field is an example. Use it for "who is my reporting manager" style designs.

Two more you should recognise: **self-relationship** (an object looking up to itself, like Account's Parent Account) and **external lookup** (pointing to external data through Salesforce Connect).

2.4.5 Comparison table

Feature	Lookup	Master-Detail	Many-to-Many (junction)	Hierarchical
Field required	Optional (can be made required)	Always	Both M-D fields required	Optional
Parent deleted	Clear value or block delete	Children deleted	Junction records deleted	N/A
Child owner and sharing	Own owner and sharing	Inherited from master	Depends on both masters	N/A
Roll-up summary on parent	No (use Flow or Apex)	Yes	Yes, on both parents	No
Where available	Any object	Custom detail objects	Custom junction object	User only
Max per object	Up to 40 relationship fields per object, at most 2 of them master-detail			

2.4.6 Querying relationships with SOQL

```
// Child to parent: dot notation with __r
SELECT Name, Student__r.Name, Batch__r.Name, Batch__r.Start_Date__c
FROM Enrolment__c
WHERE Status__c = 'Active'

// Parent to child: subquery using the child relationship name (plural)
SELECT Name, (SELECT Student__r.Name, Fee_Paid__c FROM Enrolments__r)
FROM Batch__c
WHERE Start_Date__c = THIS_MONTH
```

```
$ sf data query --query "SELECT Name, Total_Students__c FROM Batch__c" --target-org dev
 NAME            TOTAL_STUDENTS__C
 ──────────────  ─────────────────
 JEE Morning 2026 42
 NEET Evening     38
Total number of records retrieved: 2.
```

🎯 **INTERVIEW CORNER**

Q: Can you convert a lookup to a master-detail? **A:** Yes, but only if every existing child record has a value in the lookup field. Otherwise fill the blanks first with a data load.

Q: Can you convert master-detail to lookup? **A:** Yes, if there are no roll-up summary fields on the master that use it. Delete those first.

Q: Why can't Account be the detail in a master-detail? **A:** Standard objects generally cannot be on the detail side of a master-detail relationship; they can be masters. Use a lookup instead.

Q: How do you get a roll-up over a lookup relationship? **A:** With a record-triggered flow or an Apex trigger that recalculates the parent total when children are inserted, updated, deleted or undeleted.

✅ **CHECK YOUR PROGRESS 2.4**

1. **How many master-detail relationships can one object have?**
   Answer: Two.
2. **Which object supports hierarchical relationships?**
   Answer: Only the User object.
3. **What happens to detail records when the master is deleted?**
   Answer: They are deleted too (cascade delete), and restored if the master is undeleted from the Recycle Bin.

2.5 Roll-Up Summary and Formula Fields

2.5.1 Roll-up summary fields

Defined on the master, they calculate COUNT, SUM, MIN or MAX over detail records, optionally with filter criteria. Example on `Batch__c`: `Total_Students__c` = COUNT of Enrolments where Status equals Active. They update automatically when details change. Limits: by default 25 roll-up summary fields per object (can be raised, check the docs for your org). You cannot roll up a formula field that uses cross-object references or functions like `TODAY()`.

2.5.2 Formula fields

Formula fields are read-only and calculated when the record is read. Return types include Text, Number, Currency, Percent, Date, Date/Time and Checkbox. A formula can be up to 3,900 characters and its compiled size up to 5,000 bytes.

```
/* Fee_Due__c (Currency) on Enrolment__c */
Batch__r.Course_Fee__c - Fee_Paid__c

/* Payment_Status__c (Text) */
IF(Fee_Paid__c ≥ Batch__r.Course_Fee__c, "Paid",
   IF(Fee_Paid__c > 0, "Partial", "Unpaid"))

/* Age__c (Number) on Student__c */
FLOOR((TODAY() - Date_of_Birth__c) / 365.2425)

/* Grade_Band__c (Text) using CASE on a picklist */
CASE(Grade__c, "11", "Senior", "12", "Senior", "Junior")

/* Status flag with an image */
IF(Fee_Due__c > 0,
   IMAGE("/img/samples/flag_red.gif", "Due"),
   IMAGE("/img/samples/flag_green.gif", "Clear"))
```

`Batch__r.Course_Fee__c` is a **cross-object formula**: it reads a field from the parent. You can go up to 10 relationships deep across a formula. For picklists, use `ISPICKVAL(Status__c, "Active")` or `TEXT(Status__c)`, because a picklist is not plain text.

⚠️ **COMMON MISTAKE**

Using a formula field with `TODAY()` and expecting a flow or trigger to fire when its value changes. Formula values are calculated on read, not saved, so no record update happens at midnight. Use a scheduled flow or a scheduled path for time-based actions.

😄 **FUNNY SIDE**

Every admin writes a nested IF with seven levels at least once. Two weeks later, they open it, cannot understand it, and say "who wrote this?" The answer is always "you". Use `CASE()` and comments (`/* ... */`) to save your future self.

✅ **CHECK YOUR PROGRESS 2.5**

1. **Name the four roll-up summary operations.**
   Answer: COUNT, SUM, MIN and MAX.
2. **How do you compare a picklist value in a formula?**
   Answer: With `ISPICKVAL(field, "value")`, or convert using `TEXT(field)`.

2.6 Record Types, Page Layouts and Lightning Record Pages

2.6.1 Record types

Record types let one object serve different business processes. For example, `Batch__c` could have "Classroom" and "Online" record types. Each record type can show different **picklist values** and a different **page layout**. On Opportunity, Case, Lead and Solution, record types also link to **business**

processes (sales process, support process, lead process) that control stage or status values. Users get record types through profiles or permission sets.

2.6.2 Page layouts

A page layout controls which fields, sections, related lists and buttons appear on the record detail, and which fields are read-only or required *on the page*. Layouts are assigned by profile and record type combination.

> 🔑 **KEY POINT**
>
> Hiding a field on a page layout is **not security**. The user can still see it in reports, list views, the API and search. To truly hide a field, use field-level security (Chapter 3).

2.6.3 Lightning record pages and Dynamic Forms

A **Lightning record page**, built in the **Lightning App Builder**, controls the overall page: tabs, components, related lists, path, custom LWCs. With **Dynamic Forms**, you can place fields and sections directly on the Lightning page and show or hide them with visibility rules (for example, show "Loss Reason" only when Stage is Closed Lost). Lightning pages can be assigned per org default, per app, per record type and profile.

2.6.4 Schema Builder and list views

Schema Builder (Setup, Schema Builder) shows objects and relationships as a drag-and-drop diagram. It is great for explaining a design in an interview screenshot and for creating objects and fields quickly, though you still set field-level security separately. **List views** are saved filters on a tab, like "My Active Students" or "Batches Starting This Month". They can be private, shared with groups, or public, and support inline editing.

> 🎯 **INTERVIEW CORNER**
>
> **Q:** What is the difference between a page layout and a record type? **A:** A record type decides which business process and picklist values apply to a record. The page layout decides which fields and related lists are shown. Assignment is per profile and record type.
>
> **Q:** A field must be required only for one team. Where do you set it? **A:** On that team's page layout or with a Dynamic Forms required setting, or with a validation rule checking a custom permission. A field set as required at field level is required everywhere, including the API.
>
> **Q:** What are Dynamic Forms? **A:** A Lightning App Builder feature that places fields and sections directly on the record page with conditional visibility, instead of using one big page-layout block.

HANDS-ON LAB 2.1

1. In your Developer org, open Setup, Object Manager, Create, Custom Object. Label **Batch**, Record Name "Batch Name" (Text). Tick "Launch New Custom Tab Wizard". Save.
2. Add fields: Start_Date__c (Date), Course_Fee__c (Currency 8,2), Mode__c (Picklist: Classroom, Online).
3. Create record types "Classroom" and "Online" on Batch. Create two page layouts and assign them.
4. Open a Batch record, click the gear, **Edit Page**. Click "Upgrade Now" to migrate to Dynamic Forms. Add a visibility rule to show "Course_Fee__c" only when Mode is Classroom (just for practice). Save and activate.
5. Create a list view "Batches Starting This Month" with filter Start Date equals THIS MONTH.
6. Retrieve the object to your repo: `sf project retrieve start --metadata CustomObject:Batch__c`, and commit.

CHECK YOUR PROGRESS 2.6

1. **Is hiding a field on a page layout secure?**

   Answer: No. Users can still access it through reports, list views and the API. Use field-level security.

2. **Which tool builds Lightning record pages?**

   Answer: Lightning App Builder.

2.7 Designing a Data Model: A Coaching Institute

"Sharma Classes" runs JEE and NEET coaching in three centres. They want to track students, courses, batches, enrolments, fee payments and attendance. Let us design it the way you would explain it in an interview.

2.7.1 Step 1: List the nouns

Read the requirements and underline nouns: Centre, Course, Batch, Student, Enrolment, Payment, Attendance, Faculty. Each noun is a candidate object.

2.7.2 Step 2: Reuse standard objects where they fit

Students and parents are people, so use **Contact** (or Person Accounts if enabled) for students; this gives you email, phone, activities and campaigns for free. Faculty are Salesforce users, so use **User**. Parents enquiring first are **Leads**. For a simpler learning version, a custom `Student__c` is also fine; say the trade-off out loud in an interview.

2.7.3 Step 3: Decide cardinality and relationship type

Relationship	Cardinality	Type	Reason
Course to Batch	1 to many	Lookup	Batch may need its own owner (centre head); deleting a course should not wipe history
Centre to Batch	1 to many	Lookup	Centre-level sharing handled via roles
Student and Batch	Many to many	Junction `Enrolment__c` with 2 master-details	Need roll-ups (students per batch, batches per student)
Enrolment to Payment	1 to many	Lookup (required, block delete)	A junction object cannot be the master of another master-detail; total paid is summed by a flow
Enrolment to Attendance	1 to many	Lookup	Same junction rule; attendance gets its own owner
Batch to Faculty (User)	Many to 1	Lookup to User	Simple pointer

Note: `Enrolment__c` is the detail of both Student and Batch. Because it is a junction object, Salesforce does not allow it to be the **master** of another master-detail relationship. So Payment uses a required lookup, and the "total paid" figure comes from a record-triggered flow (Chapter 4) instead of a roll-up summary.

2.7.4 Step 4: Fields and calculations

- `Batch__c`: Start_Date__c, Course_Fee__c, Capacity__c, Total_Students__c (roll-up COUNT of active Enrolments), Seats_Left__c (formula: `Capacity__c - Total_Students__c`).
- `Enrolment__c`: Status__c (picklist), Total_Paid__c (Currency, kept up to date by a record-triggered flow on Payment), Fee_Due__c (formula), auto-number name `ENR-{00000}`.
- `Payment__c`: Amount__c, Paid_On__c, Mode__c (UPI, Card, Cash, Bank Transfer), Receipt_No__c (External ID, Unique).

2.7.5 Step 5: Draw it and check with reports

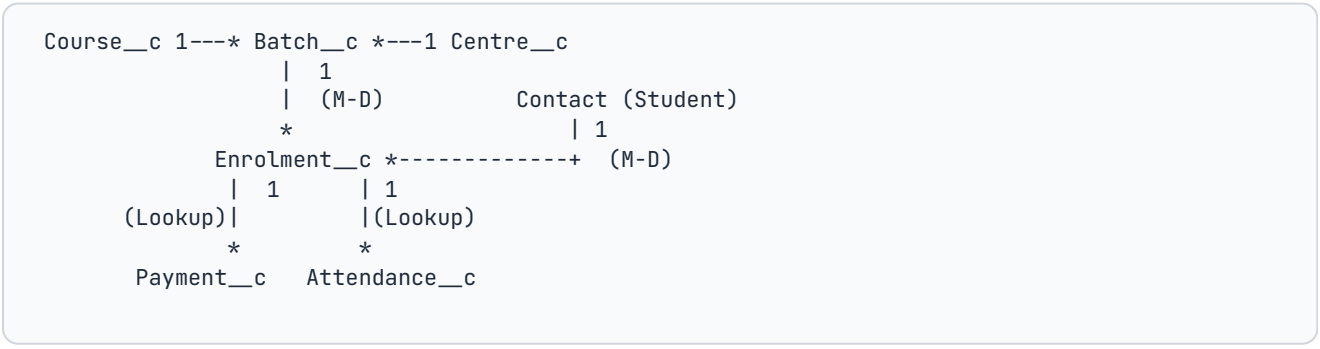


Figure 2.2: Sharma Classes data model

Test the design by asking report questions. "Fees due by centre this month?" Payment to Enrolment to Batch to Centre: yes, reachable. "Attendance percentage per student?" Attendance to Enrolment to Student: yes. If a key report cannot be built, the model needs a fix now, not after go-live.

⚠️ **COMMON MISTAKE**

Two rules trip up candidates. First, standard objects such as Contact can be the *master* in a master-detail but never the *detail*, so `Enrolment__c` with master-details to Contact and Batch is valid. Second, a junction object cannot be the master of another master-detail, so do not try to hang Payments under Enrolment with master-detail. Say both rules in an interview and you sound like someone who has actually built this.

💼 **AT WORK**

Six months after go-live, the institute asks for "reassign a student to another batch". Users get the error *"You cannot change the Batch on this Enrolment"*. Cause: master-detail reparenting is off. Fix: in the field settings for `Enrolment__c.Batch__c`, tick "Allow reparenting", deploy via change set or the sf CLI, and add a validation rule so only the centre manager (custom permission) can change it. Total time: one sprint ticket, no data loss.

🧪 **HANDS-ON LAB 2.2**

1. Create `Course__c`, `Enrolment__c` and `Payment__c` objects (you already have `Batch__c`).
2. On Enrolment, add master-detail to Contact (label Student) and master-detail to Batch. Add a required lookup from Payment to Enrolment (try master-detail first and note the error you get).
3. Create the roll-up `Total_Students__c` on Batch, a Currency field `Total_Paid__c` on Enrolment (we automate it in Chapter 4), and the formulas `Fee_Due__c` and `Seats_Left__c`.
4. Open Schema Builder, select the four objects plus Contact, arrange them and take a screenshot for your README.
5. Insert sample data: `sf data create record --sobject Batch__c --values "Name='JEE Morning 2026' Course_Fee__c=85000 Capacity__c=50"`.
6. Run the parent-to-child SOQL from section 2.4.6 and check the numbers.

✅ **CHECK YOUR PROGRESS 2.7**

1. **Why is Enrolment a junction object?**

   Answer: Because Student and Batch have a many-to-many relationship; Enrolment holds two master-details to connect them and stores enrolment-specific data.

2. **How do you verify a data model before building?**

   Answer: Ask the key report questions and check that each report path through the relationships exists.

2.8 Troubleshooting the Data Model

Symptom	Cause	Fix
Master-Detail option greyed out or error when converting lookup	Existing child records have blank lookup values	Populate the field on all records with a data load, then convert
Cannot change field type: "field is referenced by..."	Formula, flow, Apex or layout references the field	Click "Where is this used?" on the field, update references, then change
Roll-Up Summary type not available	The relationship is a lookup	Convert to master-detail if suitable, or use a record-triggered flow
SOQL error: "Didn't understand relationship 'Enrolment__r'"	Wrong child relationship name in a subquery	Check the Child Relationship Name on the master-detail field (usually plural, e.g. `Enrolments__r`)
User cannot see a new field	Field-level security not granted	Grant read/edit in a permission set; page layout alone is not enough
Formula error: "Field Status__c is a picklist field. Picklist fields are only supported in certain functions."	Picklist compared like text	Use `ISPICKVAL()` or `TEXT()`

✅ **CHECK YOUR PROGRESS 2.8**

1. **How do you find what references a field before changing it?**
   Answer: Use the "Where is this used?" button on the field's detail page in Object Manager.
2. **Why might a user not see a field you added to the layout?**
   Answer: They lack field-level security read access on it.

What Have We Learnt?

- An org is one Salesforce instance; production holds live data, sandboxes (Developer, Developer Pro, Partial Copy, Full) are for building and testing.
- Standard objects ship with Salesforce; custom objects and fields end in `__c`, and relationships in SOQL use `__r`.
- Choosing the right data type early avoids painful conversions; use External IDs for integrations and upserts.
- Lookup is loose (own owner, optional); master-detail is tight (required, cascade delete, inherited sharing, roll-ups).
- Many-to-many uses a junction object with two master-details; hierarchical relationships exist only on User.
- Roll-up summaries compute COUNT, SUM, MIN, MAX on master-detail parents; formula fields are calculated on read.
- Record types drive business processes and picklists; page layouts control fields shown; neither is security.
- Lightning App Builder and Dynamic Forms build flexible record pages with visibility rules.
- Design data models by listing nouns, reusing standard objects, deciding cardinality and testing with report questions.

Key terms: org, edition, sandbox, scratch org, standard object, custom object, API name, record ID, External ID, lookup, master-detail, junction object, hierarchical relationship, roll-up summary, formula field, cross-object formula, record type, page layout, Lightning record page, Dynamic Forms, Schema Builder, list view.

🔑 **RESUME LINE**

"Designed a 7-object data model (junction objects, roll-ups, cross-object formulas) for a 3-centre coaching institute, replacing 12 spreadsheets and cutting monthly fee-reconciliation time from 2 days to 2 hours."
"Loaded 1,200 student records via sf CLI bulk upsert on an External ID with zero duplicates."

Certification focus: this chapter is tested in the **Administrator** and **Platform App Builder** exams (data modelling is one of the largest sections) and appears in **Platform Developer I** through SOQL relationships.

Exercises

A. Multiple Choice Questions

1. Which sandbox type copies all production data?
   a. Developer
   b. Developer Pro
   c. Partial Copy
   d. Full

2. Roll-up summary fields are available on:
   a. Any lookup parent
   b. Master-detail master objects
   c. Only standard objects
   d. Only the User object

3. A hierarchical relationship can be created on:
   a. Account
   b. Any custom object
   c. User
   d. Contact

4. What happens to detail records when the master is deleted?
   a. Nothing
   b. The master field is cleared
   c. They are deleted
   d. Deletion is blocked always

5. Which formula correctly checks a picklist?
   a. `Status__c = "Active"`
   b. `ISPICKVAL(Status__c, "Active")`
   c. `CONTAINS(Status__c, "Active")`
   d. `Status__c == 'Active'`

6. Hiding a field on the page layout:
   a. Removes access in the API
   b. Is the same as field-level security
   c. Only changes what appears on that page
   d. Deletes the field data

B. Short Answer Questions

1. List four differences between lookup and master-detail relationships.
2. What is a junction object? Give an example outside education.
3. When would you use a record type instead of a new object?
4. Write a SOQL query to fetch each Batch with the names of its enrolled students.

C. Scenario Questions

1. Production is live. A manager says the "Total Students" number on batches is wrong after a bulk data load. What do you check?
2. A deployment fails with "Cannot change type of field Score__c: referenced by formula". How do you proceed?
3. Users in the Pune centre report that a new field "Scholarship %" is missing from Enrolment records, but you see it. What do you do?

D. Mini Project

Build the Sharma Classes data model (Course, Batch, Enrolment, Payment, Attendance plus Contact) in your Developer org with the roll-ups and formulas from section 2.7. Retrieve the metadata into your `sf-journey` repo, add a Schema Builder screenshot and three SOQL queries to the README, and push to GitHub.

Answer Key: Chapter 2

A. MCQ

1. (d) Full sandboxes copy all data.
2. (b) Roll-ups need master-detail.
3. (c) Hierarchical relationships are User-only.
4. (c) Cascade delete removes the details.
5. (b) Picklists need `ISPICKVAL()` or `TEXT()`.
6. (c) Layouts are presentation, not security.

B. Short Answers

1. Master-detail field is always required, lookup optional; master-detail cascades deletes; detail inherits owner and sharing from the master; only master-detail supports roll-up summaries. Also, max two master-details per object.
2. A custom object with two master-detail fields linking two objects in a many-to-many relationship. Example: `Order_Line__c` joining Order and Product, or `Assignment__c` joining Employee and Project.
3. When the records share most fields and the same security, but need different processes, picklist values or layouts, such as Classroom vs Online batches. Use a new object when fields, security and lifecycle are very different.
4. `SELECT Name, (SELECT Student__r.Name FROM Enrolments__r) FROM Batch__c`

C. Scenario Questions

1. (1) Check the roll-up filter: is it counting only Status = Active, and did the load set Status? (2) Query enrolments for one batch and compare counts. (3) Check if records were loaded against the wrong batch IDs (compare External IDs). (4) If the roll-up definition changed recently, Salesforce recalculates in the background; wait or check status. (5) Fix data with a corrected upsert and confirm with a report.
2. (1) Open the field, click "Where is this used?" (2) Update or temporarily remove the formula reference in the same deployment or a prior one. (3) Change the type in a sandbox, run tests, check reports and flows. (4) Deploy the formula change and type change in the right order, then restore the formula if needed.
3. (1) Log in as a Pune user (Setup, Users, Login) or check their permission sets. (2) The field is probably missing field-level security for their profile or permission sets. (3) Grant Read (and Edit if needed) via a permission set assigned to the Pune team. (4) Also check their page layout and record type assignment. (5) Confirm with the user.

D. Mini Project: Model Solution

Key files: `force-app/main/default/objects/Batch__c/fields/Total_Students__c.field-meta.xml` (type Summary, summaryOperation count, summaryForeignKey `Enrolment__c.Batch__c`), `Enrolment__c/fields/Batch__c.field-meta.xml` (type MasterDetail, relationshipName Enrolments), `Enrolment__c/fields/Fee_Due__c.field-meta.xml` (formula). Retrieve with `sf project retrieve start --metadata CustomObject:Course__c CustomObject:Batch__c CustomObject:Enrolment__c CustomObject:Payment__c CustomObject:Attendance__c`. README:

problem statement, Schema Builder image, relationship table from 2.7.3, the SOQL queries and a deploy section with `sf project deploy start --source-dir force-app`.