Data项目开发时间轴

先说结论，再展开点细节：

> **在有 2–3 个熟悉 AWS 的 Data/SDE 的前提下：**
>
> - **能跑通 end-to-end 的教学版 / MVP：6–8 周**
> - **做成相对“可上线”的生产级版本：3–4 个月**

当然这是粗估，只能给个量级，你可以用来规划课程 / 项目节奏。

先讲几个关键假设（Assumptions）

我下面的估算都默认：

- 团队：**2–3 人**（混合 SDE + DE），不是一个人单刷
- 技能：对 **AWS 基础、Python、Terraform** 都不陌生（哪怕不是 expert）
- 范围：
  - 支持 **834 + 837 + 835** 的**基本字段解析 + 标准化**
  - 有一个可用的 **React 上传 Portal**
  - S3 Raw / Silver / Gold + Glue Catalog + Redshift + QuickSight 都能 work
  - 有最基本的 **安全 + 审计 + 监控**

如果是一个人搞，所有时间乘个 1.5–2 倍比较现实。

按模块拆时间（可以理解为甘特图粗粒度版）

Phase 0 – 基础设施 / IaC（1–2 周）

目标：
Terraform 把最基本的东西拉起来：VPC、S3、KMS、IAM、CloudTrail、基础 Glue Catalog skeleton。

工作内容：

- 用 Terraform 定义：
  - VPC + Subnet + Endpoint（让 Glue/Lambda/Redshift 走内网）
  - S3 Raw / Lake bucket + KMS + Bucket Policy

- IAM Role 基础版本（etl / analyst / app）
- CloudTrail / CloudWatch 基本日志

- CI/CD pipeline（GitLab/GitHub Actions）打通：
  - `terraform plan` + `apply` 自动化

> 如果 IaC 模块你已经有模板（之前项目复用），这块可以压缩到 **1 周内**。

Phase 1 – 文件上传 Portal + Raw Ingestion（2–3 周）

目标：
上游可以借助 Web Portal 或 SFTP 把 CSV/X12 安全地扔进 S3 Raw，并有 basic metadata。

工作内容：

- 后端：
  - API Gateway + Lambda
  - `POST /upload-request` → 返回 pre-signed URL
  - DynamoDB 记录 file metadata（file_type, uploader, checksum, status）
- 前端：
  - React + TypeScript + 简单 UI（登录、选择文件、上传、显示历史记录）
  - 调 Cognito 登录 / 调 API Gateway
- 集成：
  - S3 Event → Lambda → 写 SQS/Step Functions，后续 ETL 用

时间估：

- 熟练的前后端 + AWS 工程师：**2 周够用**
- 如果团队里没有前端，会卡一点，可能拖到 **3 周**

Phase 2 – X12 → CSV 解析 + Silver 标准化（最费时间，3–5 周）

目标：
把 834 / 837 / 835 从文件级拆成 Member / Claim / Claim Line / Payment 等实体，落到 Silver 层，并有基础数据质量校验。

工作内容：

1. **定义 Mapping & Schema（业务 + 技术一起）**
   - 834: Member / Coverage / Enrollment Event

- 837: Claim Header / Claim Line / Diagnosis / Provider
- 835: Payment / Adjustment / Denial Reason

2. **实现解析 Job**
   - Python / PySpark / Glue Job
   - 读 Raw CSV（或 X12，如果要支持直接解析 X12）→ 输出标准化 CSV/Parquet
3. **数据校验 & Error Handling**
   - 必填字段检查
   - PK/FK 合法性（member_id, claim_id 等）
   - 失败记录写入 `_error` path 或单独 error 表

时间估：

- 如果只做：
  - **先实现 1 种（比如 837 Claim）→ 2 周左右**
  - 再复制到 834 + 835（重用框架）→ 再加 1–3 周
- 加起来 **3–5 周** 比较正常
  （这块是“脑壳最痛”的地方，因为跟业务字段细节绑得很紧）

Phase 3 – Gold 模型（Redshift + 星型模型 + 基础报表）（2–3 周）

目标：
有一套可用的 **Dim / Fact** 模型，能支持典型的 Claims 分析与 Dashboard。

工作内容：

- 在 Redshift（推荐 Serverless）里：
  - 设计并建：
    - `dim_member / dim_provider / dim_plan / dim_diagnosis / dim_facility`
    - `fact_claim / fact_claim_line / fact_payment / fact_eligibility`
  - 写 ETL SQL / Glue Job 把 Silver → Gold
- DQ（简单但必要的）：
  - record count 对齐
  - 金额 sum 对比（raw vs silver vs gold）
- QuickSight：
  - 几个基础 dashboard：

        - Claim Cost 分布
        - Provider Performance
        - Denial / Adjustment 统计
        - Member Risk / High Cost Claim 分布

时间估：

- 模型 + ETL SQL：**1–2 周**
- QuickSight dashboard + 调优：**1 周**

Phase 4 – 安全加固 + 审计 + 文档（1–2 周）

目标：
补完 HIPAA 视角的“安全闭环”和团队内部可维护性。

工作内容：

- IAM + Lake Formation 权限细化：
    - ETL 角色 vs 分析角色 vs ops 角色
- Macie（可选）跑一遍 PII 扫描，验证配置
- CloudWatch Dashboard：
    - pipeline 延迟、失败率、文件处理量
- 文档：
    - 架构图（一个 PPT / draw.io）
    - Runbook（出问题怎么排查）

时间估：

- 如果是教学/内部 demo，**1 周**
- 如果要做得更严谨一点，**1–2 周**

综合一下时间线（可并行 + 有重叠）

大概可以这样看：

- **Week 1–2**
    - IaC 基础 + 部分 Security 搞定
    - 同时可以起 Portal 的框架
- **Week 3–4**

    - 上传 Portal 完成 + Raw Ingestion 打通
    - 开始 837 → Silver 的解析
- **Week 5–6**
    - 834 / 835 解析补齐
    - 初版 Gold 模型 + Redshift 打通
- **Week 7–8**
    - QuickSight Dashboard + DQ
    - 安全 / 审计 / 文档收尾

所以：

- **教学版 / PoC：**
    - 如果只做 837 + Portal + 简化安全（不搞太复杂的权限），**4–6 周** 就可以看到很完整的 Demo。
- **比较完整的 Lakehouse + 三种 EDI + 安全 + Dashboard：**
    - **6–8 周** 是比较稳的估计（2–3 人团队）
- **真正生产水准（要过内部安全评审 + OPS 标准）：**
    - 引入更多自动化测试、DQ 框架、复杂权限、审计报表 → **3–4 个月** 合理。