1. Job Scheduler System

📌 核心要求

- Create, schedule, and run jobs
- Report failures and successes
- At-least-once task execution guarantee
- View running/completed job statuses
- Handle SLA violations (jobs taking longer than expected)

🧩 必答问题

1. 增加任务优先级功能如何改进系统？
2. 如何查询 recurrent jobs 的状态？如何支持 cron jobs？如何将其转化为普通 jobs？
3. 多台机器如何并发 poll jobs？
4. 如何控制 consumers 数量？如果 consumer 执行中挂了怎么办？
5. 选择 NoSQL 的理由？SQL 一次 poll 10,000 jobs 可行吗？
6. Job execution 超时怎么办？Kill job？Alert？
7. 若用 coordination service（如 ZooKeeper）协调 scheduler，数据结构如何设计？如何存储关系？如何实现 at-most-once execution？
8. Retry 和队列管理策略如何优化，尤其在支持任务优先级的场景下？

✅ 2. Stock Broker / Limit Order System

📌 核心题目

设计券商限价单系统，实现用户发单后验证余额并通过 API 向市场提交

🔧 系统能力要求

- 限价单处理流程（含 order expiration、并发控制、会计规则）
- 高并发订单支持
- 与市场的 API 通信及错误/故障恢复机制
- 限价单过期逻辑（每日清除）
- 数据一致性和 fault tolerance
- 防止重复提交
- 用户账户余额验证

🧩 高频考点

- 如何处理 NYSE/Nasdaq error/failover？
- 如何确保同一订单不被重复提交？
- 如何验证用户 account balance？
- 限价单的过期清理逻辑
- 支持并发 & 数据一致性（事务、锁、乐观并发控制）
- 交易失败后的恢复机制
- 数据模型设计（e.g. order schema）
- 高并发下的数据隔离与校验

✅ 3. Real-Time + Historical Stock Price API

📌 核心题目

设计系统从两个 NASDAQ TCP stream ingest 数据，并提供以下 API：

- `/stockSymbol/live` — 实时价格查询（需支持 10M QPS）
- `/stockSymbol/historical?range=...` — 历史价格查询（支持 1M QPS）

🧩 高频考点

- 实时数据 ingestion：如何进行去重（dedupe）与 failover？
- 使用 Redis Cluster + Kafka 支持 100k QPS 写入
- 实时 + 历史数据存储方案（Redis + PostgreSQL or Data Lake）
- candlestick 聚合系统如何设计（OHLC 计算）
- Query 缓存 + 负载均衡（e.g. CDN / API Gateway）
- 返回全部交易数据 vs 聚合后数据

✅ 4. Compute Ads Campaign ROI System

📌 核心题目

设计广告 ROI 系统，包括：

- 实时数据收集（点击/转化事件）
- ROI 计算模块（离线批处理分析）

🧩 高频考点

- Kafka + HDFS / Data Lake 的 ingestion 架构
- 使用 Spark / MapReduce 进行 ROI 离线批处理计算
- 多源数据（点击日志 + 转化日志）如何 join？

- ROI 伪代码：`group by campaign`, `join user events`, `sum up spend vs return`
- 如何处理延迟数据 / 数据缺失（late arrival, reprocessing）
- ROI 指标展示、分 campaign 报表