作业1: sqlite、 orm、 es 的使用学习，自己构建数据插入

MySQL（以及PostgreSQL, SQL Server等）是**客户端/服务器模式**的关系型数据库。它需要一个独立的数据库服务器进程，应用程序通过网络协议（如TCP/IP）与之通信。**在公司内部的典型应用场景：**

1. **核心业务系统**：存储最重要的业务数据，如用户信息、订单、商品、交易记录等。这是MySQL最主要的工作。
2. **Web应用后端**：几乎所有的动态网站、APP后端API，其数据都存储在MySQL这类数据库中。
3. **内部管理系统**：如ERP（企业资源计划）、CRM（客户关系管理）、OA（办公自动化）等系统，通常也使用MySQL作为中央数据库。
4. **需要高并发、多用户写入的场景**：MySQL有完善的并发控制机制（如锁、事务隔离级别），能保证大量用户同时读写时的数据一致性和完整性。

Elasticsearch (ES) 的使用场景和 MySQL/SQLite 有本质的区别。 它不是用来替代关系型数据库的，而是为了解决后者不擅长的问题。它的核心优势在于：**全文搜索、日志处理、以及复杂聚合分析**。 ES 使用**倒排索引**，类似于一本书最后的“索引”页，可以瞬间找到包含某个词的所有文档。MySQL 的模糊查询则是扫描每一行数据，效率极低。

- **电商网站**：搜索商品。用户输入“红色 连衣裙 修身”，ES 可以高效地匹配所有包含这些关键词（甚至近似词）的商品，并按相关度排序。这在 MySQL 里用 `LIKE '%xxx%'` 操作会导致性能灾难。
- **内容平台**：如新闻网站、博客平台（如知乎、Medium）搜索文章内容。

MySQL基本免费，社区版功能强大，完全免费。企业版需要付费，提供额外支持、工具和功能，但大多数公司使用社区版就足够了。ES 严重依赖内存来缓存倒排索引和加速搜索。数据写入和段合并时对 CPU 和磁盘 I/O 要求也很高。为了追求极致的写入和查询速度，通常需要配置更高性能的 SSD 硬盘。

作业2 :RAG 实验代码，重新跑通。截图

系统化、可复现、高效率地探索不同想法，避免混乱。**将代码、配置、数据分离**。不要让超参数（如学习率、batch size）或文件路径硬编码在代码里。

```
my_dl_project/
├── configs/            # 存放所有配置文件（.yaml, .json）
│   ├── exp_001.yaml
│   └── exp_002.yaml
├── data/               # 存放原始数据和预处理后的数据
├── src/                # 主要源代码
│   ├── data_loading.py
│   ├── model.py
│   ├── train.py
```

```
│   └── utils.py
├── scripts/            # 运行脚本
├── experiments/        # 自动生成，存放每次实验的独有信息
│   ├── exp_001_20231027/  # 以实验名+时间命名
│   │   ├── logs/          # 训练日志，TensorBoard文件
│   │   ├── checkpoints/   # 模型权重
│   │   └── config.yaml    # 本次实验完整的配置副本
│   └── exp_002_20231028/
└── requirements.txt    # 项目依赖
```

如果你的想法包含多个组件（例如，同时改了网络结构和新加了一个损失函数），需要通过消融实验来验证每个组件的**独立贡献**。

- **Example**: `Base模型` -> `Base + 组件A` -> `Base + 组件A + 组件B`
- 如果`Base + 组件A`提升了，说明A有效；如果`Base + 组件A + 组件B`提升更多，说明B也有效。