笔试题

题目一

环境约束：

- Python ≥ **3.10**
- **只允许**使用标准库（ `sqlite3 /dataclasses /typing /datetime /contextlib` 等）
- 禁用第三方 ORM / 驱动
- 系统中需能读写本地文件

任务概览

编写 **单一脚本** `credit_app.py` ，实现一个可直接运行的本地原型，覆盖个人消费信贷业务的 **开户 → 申请 → 放款 / 拒绝 → 还款 → 报表** 全流程。脚本第一次运行时应自动生成 `credit.db` 并自建表，后续重复运行不报错。

1. 数据库模型（自动建表）

在 CreditApp.init 中打开/创建数据库，并执行 以下 DDL（包括 PRAGMA foreign_keys = ON）：

复制代码

```
CREATE TABLE IF NOT EXISTS customers (
    id          INTEGER PRIMARY KEY AUTOINCREMENT,
    name        TEXT NOT NULL,
    email       TEXT NOT NULL UNIQUE,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS loans (
    id              INTEGER PRIMARY KEY AUTOINCREMENT,
    customer_id     INTEGER NOT NULL,
    principal_cents INTEGER NOT NULL CHECK (principal_cents > 0),
    interest_rate   REAL NOT NULL CHECK (interest_rate BETWEEN 0 AND 1),
```

```
    term_months     INTEGER NOT NULL CHECK (term_months > 0),
    status          TEXT NOT NULL CHECK (status IN
('pending','approved','rejected','closed')),
    applied_at      TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    approved_at     TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS repayments (
    id           INTEGER PRIMARY KEY AUTOINCREMENT,
    loan_id      INTEGER NOT NULL,
    amount_cents INTEGER NOT NULL CHECK (amount_cents > 0),
    paid_at      TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (loan_id) REFERENCES loans(id) ON DELETE CASCADE
);
```

2. 面向对象设计

2.1 数据类（全部使用 `@dataclass` 并加类型注解）

类名	字段
`Customer`	`id` (可选)、 `name` , `email` , `created_at` (可选)
`Loan`	`id` , `customer_id` , `principal_cents` , `interest_rate` , `term_months` , `status` , `applied_at` , `approved_at`
`Repayment`	`id` , `loan_id` , `amount_cents` , `paid_at`

2.2 核心业务类 `CreditApp`

- 持有 `sqlite3.Connection`
- 构造函数负责：启用外键、自动建表
- 对外提供下列 **业务接口**（全部需使用参数化查询，并用 `with self.conn:` 保证事务原子性）

方法	说明
`add_customer(c: Customer) -> None`	若 `email` 已存在抛 `DuplicateKeyError`
`apply_loan(email: str, principal_cents: int, interest_rate: float, term_months: int) -> int`	新建 `status='pending'` 贷款，返回贷款 ID
`approve_loan(loan_id: int, approve: bool) -> None`	仅能处理 `pending` 贷款；转为 `approved` 或 `rejected`，并写 `approved_at`
`record_repayment(loan_id: int, amount_cents: int) -> None`	仅允许 `approved` 贷款；插入还款记录；若已全额还清，则把 `status` 置 `closed`
`customer_balance(email: str) -> tuple[int, int]`	返回 `(总借款本金, 已还金额)`
`overdue_loans(days: int = 30) -> list[tuple]`	列出 **距最后付款 > days 天** 且仍为 `approved` 状态的贷款：`[(loan_id, customer_name, outstanding_cents, last_payment_date)]`
`portfolio_summary() -> dict`	统计字段：`total_customers`，`active_loans`，`avg_interest_rate`，`avg_principal_cny`，`default_ratio`（逾期>90天/已批准贷款）

2.3 自定义异常

可以自定义一些自己的需要的异常

复制代码

```
class DuplicateKeyError(Exception): ...
class RecordNotFound(Exception): ...
class InvalidOperationError(Exception): ...
```

3. 代码规范

规范的命名

合适的docstring

合适的typing

自定义异常

加分项 可以考虑对代码文件对于功能进行抽取 然后设置不同的文件结构

4. 演示脚本（文件末尾）

复制代码

```
if __name__ == "__main__":
    app = CreditApp()

    # 1) 新增两位客户
    # 2) 各自申请贷款并审批
    # 3) 录入还款，故意重复审批触发异常
    # 4) 打印 customer_balance、overdue_loans、portfolio_summary
```

题目二

环境约束:

- Python ≥ **3.10**

一、 `Card` 类

1. **属性**
   - `number`：可取 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A
   - `suit`：可取 **Spade / Heart / Club / Diamond**（或简写 **S / H / C / D**）
2. **成员函数**
   a. `show()`
      - 示例输出：`S-10` 表示黑桃 10；`D-J` 表示方块 J
   b. `compare(other: Card)`
      - 若本牌点数更大，返回 `">"`
      - 若更小，返回 `"<"`
      - 点数相同，返回 `"="`
      - 比较顺序：A > K > Q > J > 10 > 9 > … > 3 > 2

二、`Deck` 类

1. `__init__()`
   - 生成 52 张 `Card`，顺序固定：
     `S-2, H-2, C-2, D-2, S-3, H-3, … , S-A, H-A, C-A, D-A`
2. `validate()`
   - 返回 `True`：牌堆包含 *恰好* 52 张合法且互不重复的 `Card`
   - 否则返回 `False`（例如：有 5 张 A，或出现两个黑桃 J）
3. `shuffle()`
   - 随机打乱牌序，但打乱后仍需满足 `validate() == True`
4. `draw()`
   - 从牌顶抽一张（无放回），返回该 `Card` 对象，并将其自牌堆移除

三、实验流程

1. **初始化** 一副牌
2. **洗牌** → 调用 `validate()`（应返回 `True`）
3. **抽一张牌**，显示该牌 → 再次 `validate()`（应返回 `False`）
4. **再抽一张牌**，显示；将其与步骤 3 的牌 **compare**
5. **重复步骤 1–4 共 1,000 次**，统计步骤 4 中 `compare` 返回 `"="` 的频率
6. 将该频率与 **理论概率** 比较；若重复次数增至 **100,000**，频率是否进一步收敛?

笔试题3

环境约束：

- Python ≥ **3.10**

目标

Your objective is to select the best fertilizer for different weather, soil conditions and crops.

你的目标是为不同的天气、土壤条件和作物选择最好的肥料

评估

Submissions are evaluated according to the Mean Average Precision @ 3 (MAP@3):

提交将根据平均精度@3 （MAP@3）进行评估：

$$MAP@5 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,5)} P(k) \times rel(k)$$

where U is the number of observations, P(k) is the precision at cutoff k, n is the number predictions per observation, and rel(k) is an indicator function equaling 1 if the item at rank k is a relevant (correct) label, zero otherwise.

其中U是观测数，P(k)是截止点k处的精度，n是每个观测预测的数，如果秩k处的项是相关（正确）标签，rel(k)是等于1的指标函数，否则为零。

Once a correct label has been scored for *an observation*, that label is no longer considered relevant for that observation, and additional predictions of that label are skipped in the calculation. For example, if the correct label is `A` for an observation, the following predictions all score an average precision of `1.0`.

一旦一个观测结果得到了正确的标签，该标签就不再被认为与该观测结果相关，在计算中也忽略了对该标签的其他预测。例如，如果一个观测的正确标签是A，那么下面所有的预测的平均精度都是1.0。

复制代码

```
[A, B, C, D, E]
[A, A, A, A, A]
[A, B, A, C, A]
```

基本要求

- 对数据进行EDA
- 特征工程
- 搭建模型得到测试集预测结果

加分项

- 使用agent辅助挖掘特征

提交文件

1. 测试集结果

对于测试集中的每个id，可以预测最多3个肥料名称值，并分隔预测结果。具体提交示例参考相关文件中的sample_submission.csv：

```
id,Fertilizer Name
750000,14-35-14 10-26-26 Urea
750000,14-35-14 10-26-26 Urea
...
```

2. EDA、特征工程、模型模块的相关jupyter、py等过程文件

相关数据

模型笔试题数据.zip
11.8MB