今天一天可算是忙坏了，刚实习结束，笑了，明天要对因子挖掘程序进行彻底的重构，淦，换nm啊（笑）

不说了，我继续针对这个综合选股速度进行速度上的进一步优化，今天实盘操作之前数据导出了，但是没有及时更新数据库，导致了直接推迟了20分钟，这是重大失误，同时发现了目前现阶段的系统仍然存在问题，那就是运行时间过长，之前已经通过了两种手段进行优化，第一是减少股票池，第二是减少策略。但是策略本身是优质的，不能被删减，只能减少大量垃圾股票，目前股票池3876只，时间缩短了五分钟。

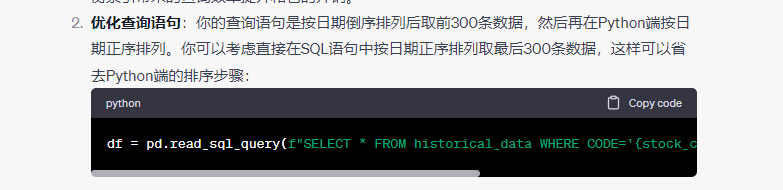
缩短后的程序跑盘时间死死控制在20分钟，很难进行底层优化（忙了一天没想到好的解决方案。）

目前尝试过的解决方案有以下几点：

1. 该系统的底层算法是多进程处理，每一只股票都会建立一次SQL链接，一开始提出的想法是，我们能不能通过一次性获得全部数据，然后对df上进行搜索，然后程序是跑通了，但是时间直接干到了三个小时。。。。（详见selectiontestV3）
2. 上面的想法被否决之后，提出了新的想法，那就是能不能节省链接？直接把预加载和sql链接外置，而不放在函数内？很明显，经过一系列报错之后，答案是显而易见的不行，原因在于多进程。多进程导致了可能存在的多个子进程都要访问这个数据库的情况，造成塞车，而不是一开始的各个进程独享自己的连接，因此此类方案也不行。

## 目前尝试的方法只能局限在查询数据库本身

1. 再次重看代码，发现：



对此种情况进行更改，查看有无效率提升。应该是有的，这相当于少一步操作。（nm出大问题，根本不能用，我想起来为什么先倒这取了，因为策略是从后往前排的，淦。新的日期排后面。而且效率根本没提升）

1. 增加索引，这应该是可能提升最大的地方。

您的程序主要是通过股票代码（CODE）查询每个股票的历史数据，然后根据日期（date）进行排序。因此，创建索引的最佳选择应该是CODE和date字段。

如果您只针对一个字段（例如CODE）创建索引，那么数据库在通过CODE查找数据时可以非常快速地定位到相关的行，但在对这些行按日期排序时，可能仍然需要执行全表扫描，这就抵消了索引带来的部分性能提升。

而如果您创建了一个CODE和date的联合索引，那么在查询特定CODE的数据并按date排序时，数据库可以直接使用索引来获取结果，无需额外的排序操作，这样可以最大程度地提高查询性能。

总的来说：

* 单独针对CODE创建索引：可以提高通过CODE查找数据的性能，但在排序时可能仍需要全表扫描。
* 单独针对date创建索引：除非您需要经常按日期查询所有股票的数据，否则这个索引可能没有太大用处。
* 针对CODE和date创建联合索引：可以同时提高通过CODE查找数据和按date排序的性能。

但请注意，索引并非越多越好。创建索引会消耗存储空间，而且在插入、更新和删除数据时，数据库需要维护索引，这可能会降低写入性能。因此，您应该根据实际情况，权衡查询性能和写入性能，来决定是否需要创建索引，以及创建哪些索引。



增加索引的代码如下: conn = sqlite3.connect(r'stocks617.db') # 连接到数据库

c = conn.cursor() # 创建一个数据库游标

# 在 historical\_data 表的 CODE 和 date 字段上创建一个索引，命名为 idx\_code\_date

c.execute("CREATE INDEX IF NOT EXISTS idx\_code\_date ON historical\_data (CODE, date)")

conn.commit() # 提交数据库事务

conn.close() # 关闭数据库连接

结果：

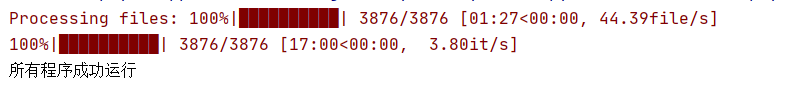
性能终于提升了，提升了三分钟。效率提升11%

更新数据库程序也有提升，现在只要1分半就能完成更新，足足提升了百分之五十的效率。

我的笔记本cpu是6核，如果核心数目翻倍，那么对程序的效率理论上来说应该也是翻倍的提升，那么跑大盘时间能缩短到10分钟以内。

我靠，我是小丑，nmd联合索引没创建成功，我是进入了数据库终端控制台才弄好的，淦！

我自己通过终端控制台创建了一个新的索引，最后效果如下：



还是有提升的，差不多三分钟，极限了。之后对于这个系统而言只能从硬件上，股票池，策略数上进行提升了。

通过语句：EXPLAIN QUERY PLAN SELECT *\** FROM historical\_data WHERE CODE='689009' ORDER BY date DESC LIMIT 300

我们确实可以看出，我创建的联合索引被使用了：

