**1.基本信息的管理：**

1）实现学生基本信息的输入、删除、修改工作，学生基本信息包括学号、姓名、性别、籍贯等

2）实现课程基本信息的输入、删除、修改工作，课程基本信息包括课程编号，课程名称、学号、学分、先修课等

3）按照特定的信息进行查找，并按照特定的要求进行排序

**2.学生成绩的管理**

1）学生成绩的输入、修改、删除等基本操作

2）成绩的查询功能，可以按照学号、姓名等进行成绩查询

3）成绩的综合统计功能，包括学生成绩按照不同条件进行排名，管理人员可以选择不同的排序条件，并可以将排序结果进行打印输出

4）计算每个学生的总成绩、平均成绩，所有学生单科的平均成绩，找出总成绩和单科成绩最高的学生并输出

5）计算各班的总分和平均分，并按照平均分将各班成绩进行排序

6）统计不及格考生的名单以及成绩

7）学生奖励统计，自动生成每班获得奖学金的同学的信息，并可以将奖学金分成不同的等级，以及所得奖学金的数目

8）实现了历史学生的查询、恢复功能，将删除的学生存放进历史学生表，并可以进行恢复，防止误删操作的发生

9）学生数据库的维护，实现数据备份以及数据恢复，防止数据丢失

10）学生信息报表输出，包括学生成绩汇总报表、需重修学生统计报表（即不及格学生）

二、课程设计的要求与数据

1.原始数据的录入（包括学生信息、课程信息和学生成绩的录入）

2.课程查询、学生信息和成绩查询

3.报表输出。主要包括学生成绩汇总报表等

4.数据维护。主要包括数据备份、数据恢复等

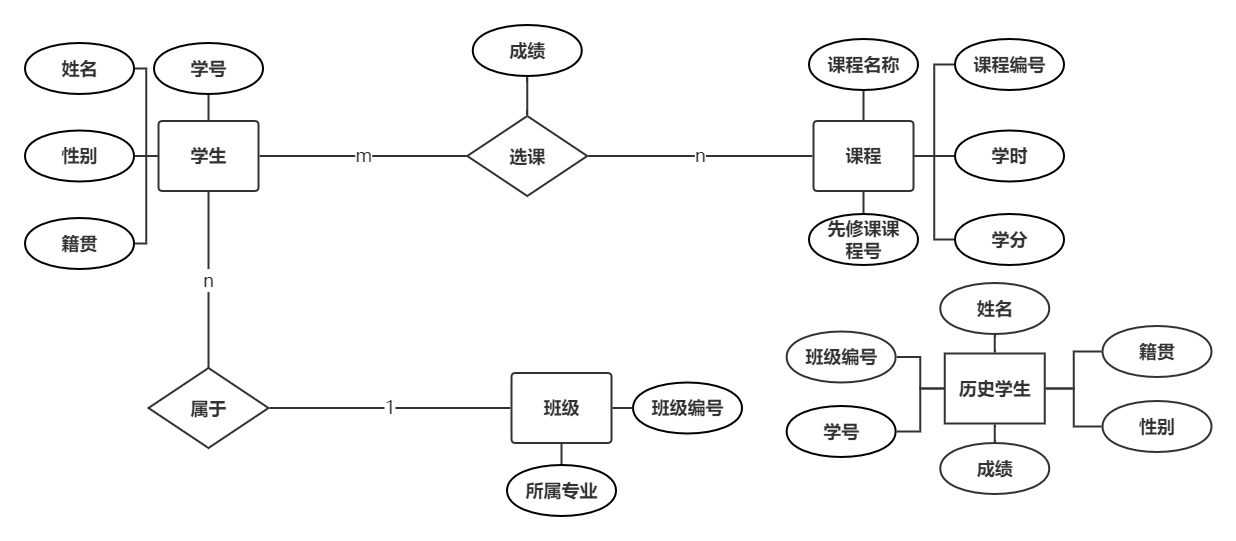
三、课程设计应完成的工作

**1.课题需求分析分析以及小组分工**

课题需求分析是课程设计一个重要环节，它准确地回答“系统必须做什么”这个问题，我们通过课程设计要求以及自身对学生管理系统的使用了解，分析现有的学生成绩管理系统，获取了此次课题系统设计的需求，即课程设计的内容。

**2.由需求分析得出的结论进行关系数据库设计，绘制E-R图，定义各关系模式的联系以及约束条件**

绘制的E-R图如下：



关系模式的联系：

学生(学号，姓名，性别，籍贯)

课程(课程编号，学时，学分，课程名称，先修课课程号)

选课(课程编号，学号，成绩)

班级(班级编号，所属专业)

属于(学号，班级编号)

历史学生(学号，姓名，性别，籍贯，班级编号，成绩)

约束条件：

alter table `选课` add constraint fk\_1 FOREIGN KEY(学号) REFERENCES `学生信息`(学号)

alter table `选课` add constraint fk\_2 FOREIGN KEY(课程编号) REFERENCES `课程`(课程编号)

alter table `属于` add constraint fk\_3 FOREIGN KEY(学号) REFERENCES `学生信息`(学号)

alter table `属于` add constraint fk\_4 FOREIGN KEY(班级编号) REFERENCES `班级`(班级编号)

alter table `学生` add primary key(学号)

alter table `课程` add primary key(课程编号)

alter table `班级` add primary key(班级编号)

alter table `历史学生` add primary key(学号)

alter table `学生` add CONSTRAINT check\_1 check(学号>0)

alter table `选课` add CONSTRAINT check\_2 check(0<=成绩<100)

**3.基本框架的建立**

通过在网上查阅了相关资料，我们小组最终定下了此次课设的基本框架，即使用python编写此次课程设计，其中python调用pyqt5设计前端页面，调用pymysql连接数据库，至于功能实现也是基于python自带库实现。

**4.通过代码实现系统**

4.1先说图形界面

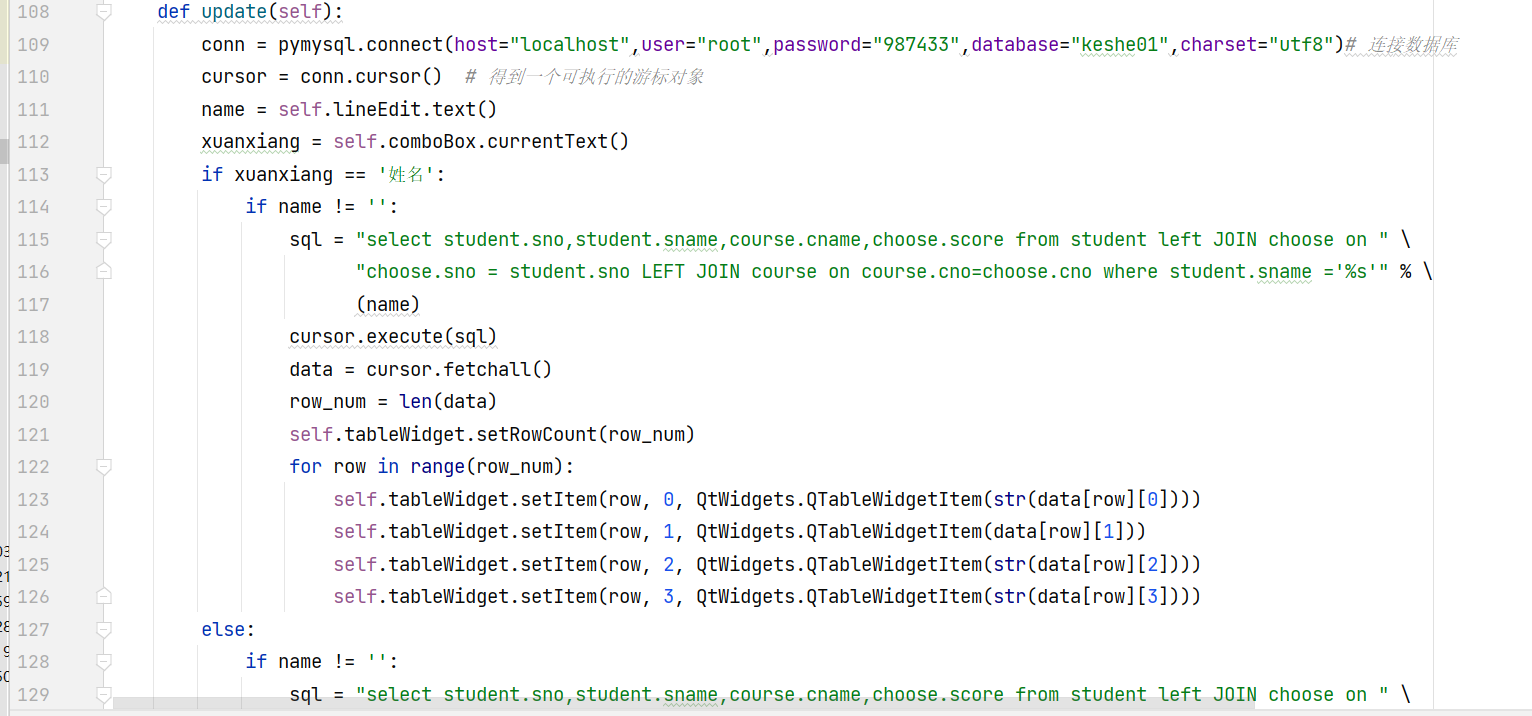


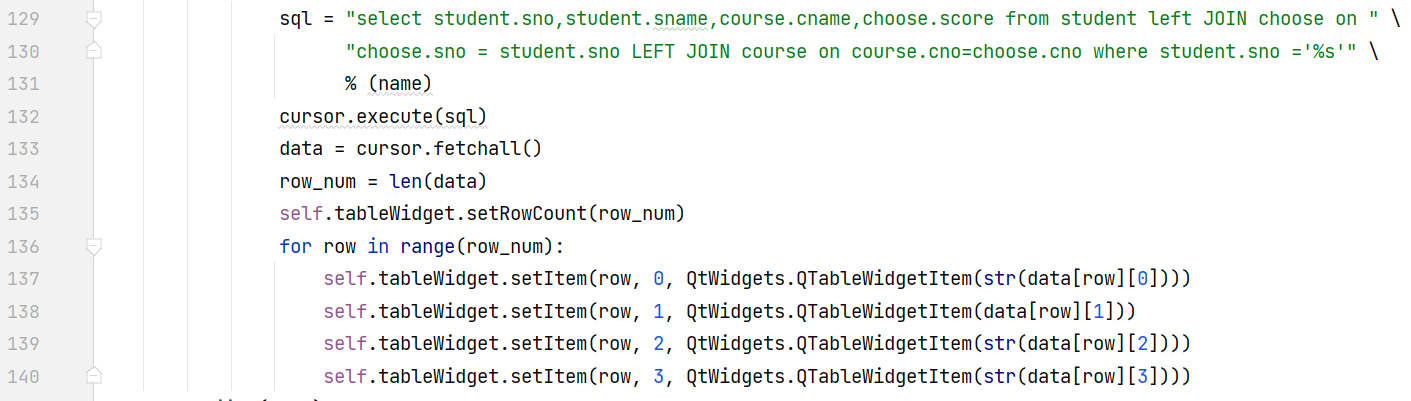
这个左右布局是比较适合学生成绩管理系统的，这也是我们从以前使用的系统中得到的灵感，具体实现代码如下：



4.2功能部分（按图形界面左边任务栏功能顺序叙述）

4.2.1首先是学生成绩的查询，可以通过学号或姓名查询，部分代码如下：



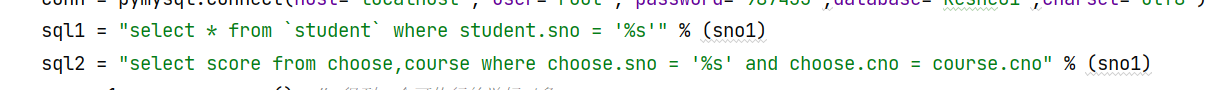


4.2.2统计不及格学生信息部分代码如下：

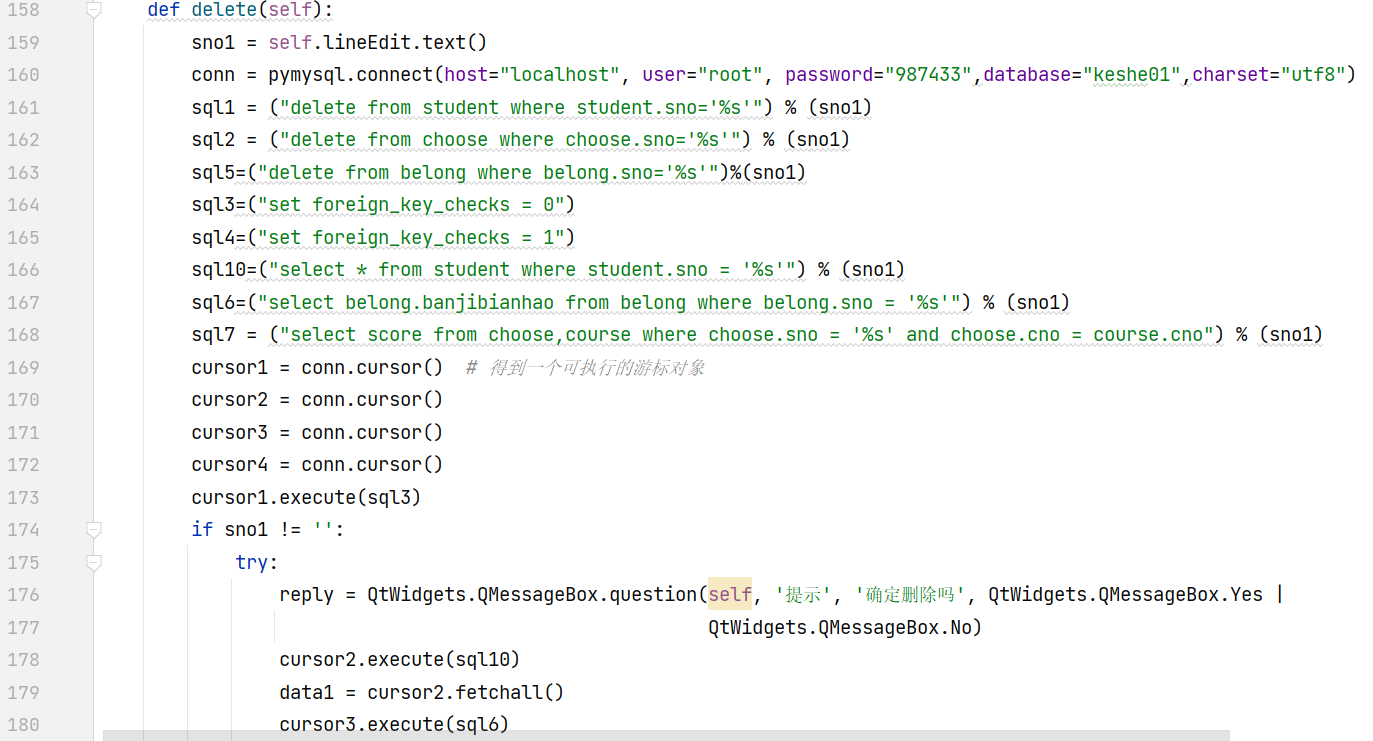


4.2.3学生信息删除，这部分我们实现通过学号查询学生信息是否存在，并将学生信息填入表格，如不存在报错，如存在，使用者可以根据查询得到的内容决定是否删除该学生，同时删除的学生信息也将会被放进历史学生中

查询功能代码类似4.2.1，但sql做出了更改，如下图：

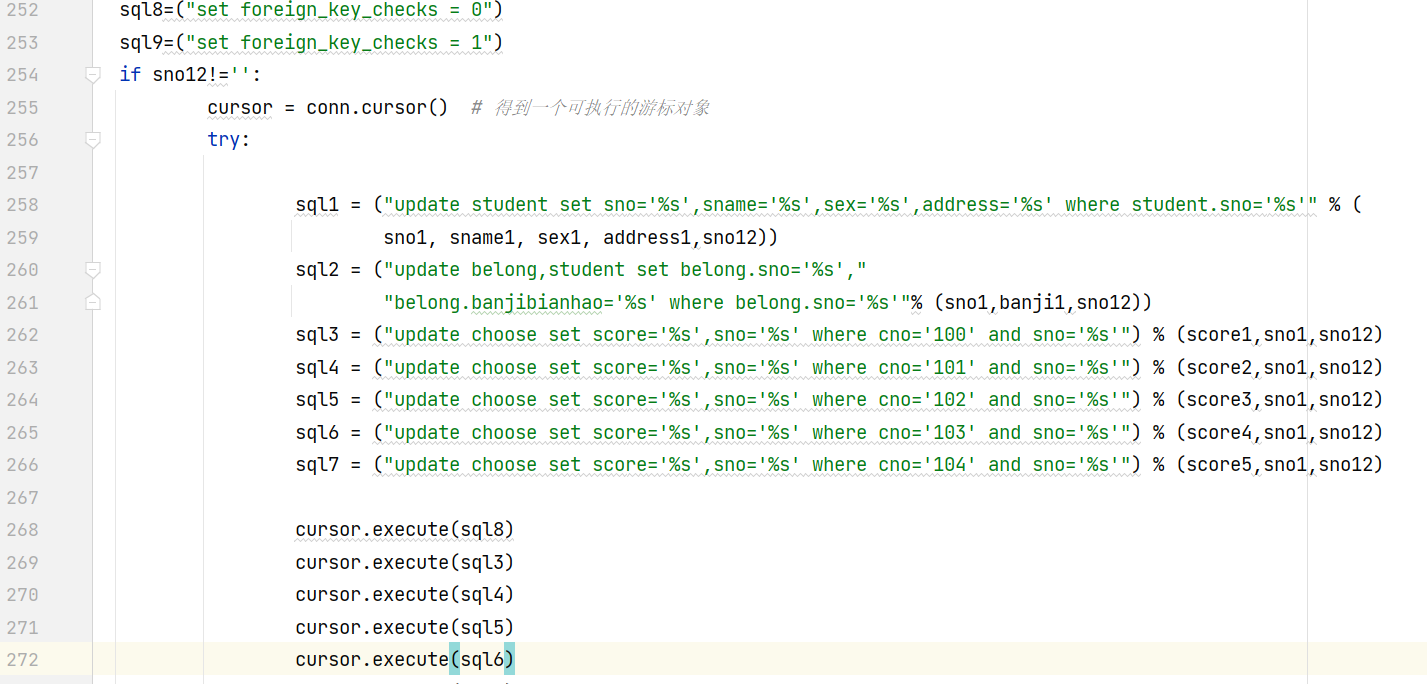


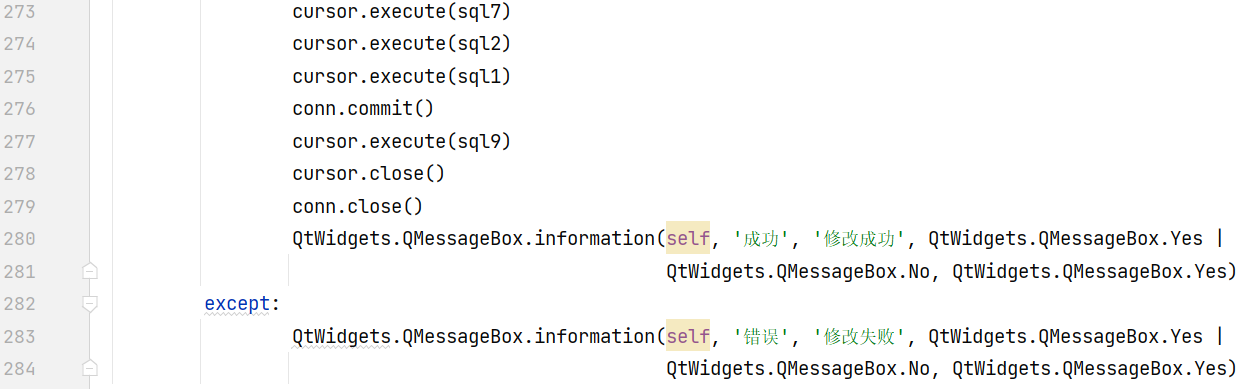
删除功能由于主键和外键的存在，因此我们在进行删除的时候需要将外键约束关闭，在删除操作结束后将外键约束打开，其中sql8在执行插入操作，将删除的人员信息插入到历史学生表中，具体代码如下：



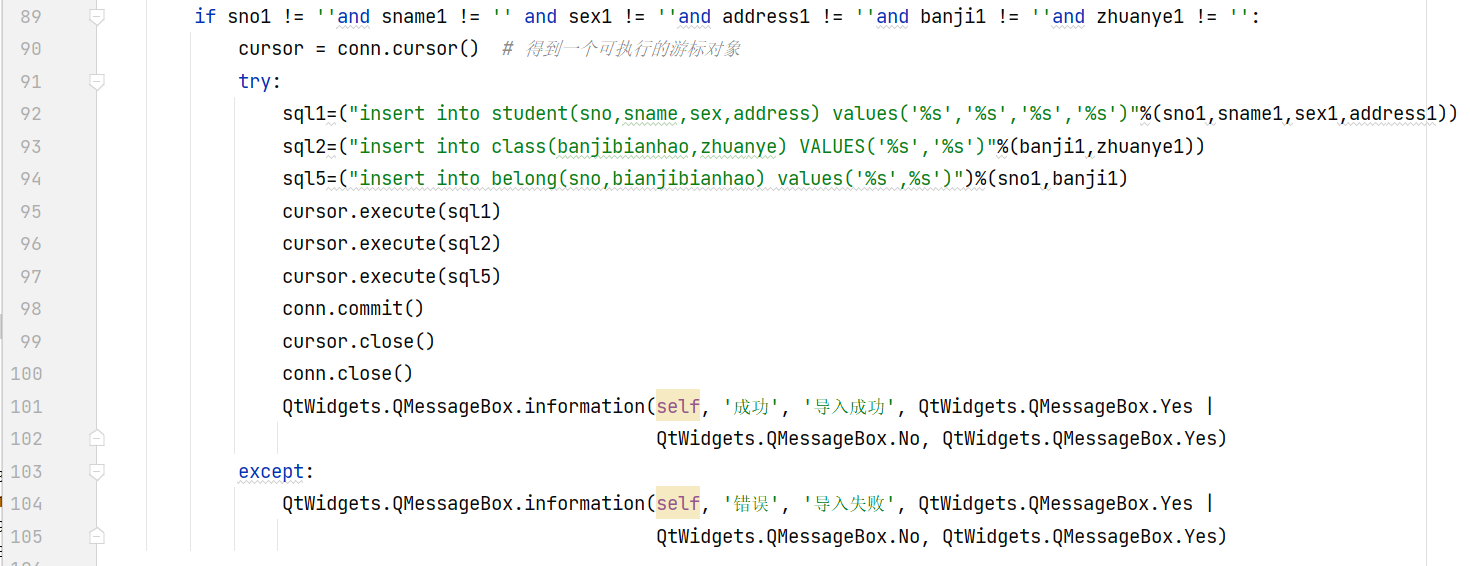


4.2.4学生信息修改，这部分同样用到了查询功能，查询功能的代码就不赘述了，而修改功能由于主键和外键约束，因此我们在进行修改同时，也要将外键约束关闭，修改操作结束后再打开，部分代码如下：

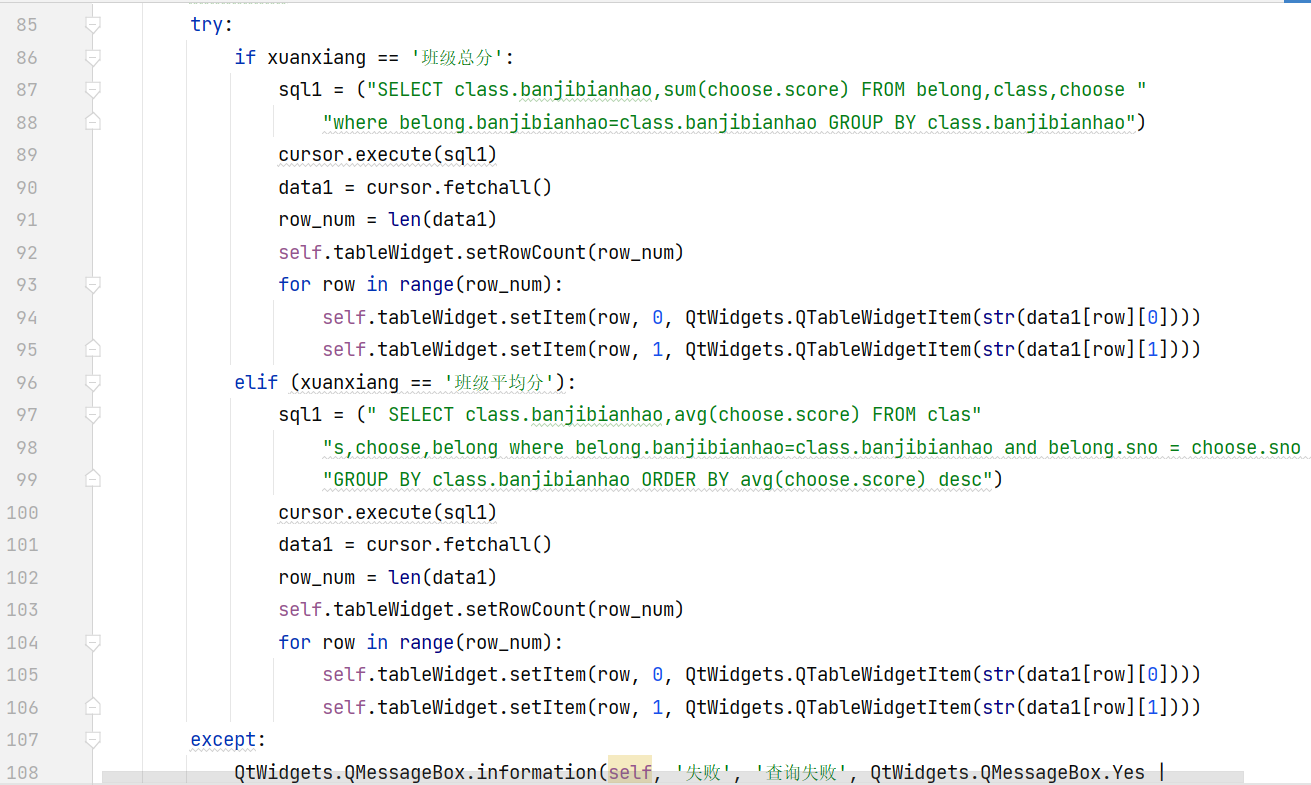




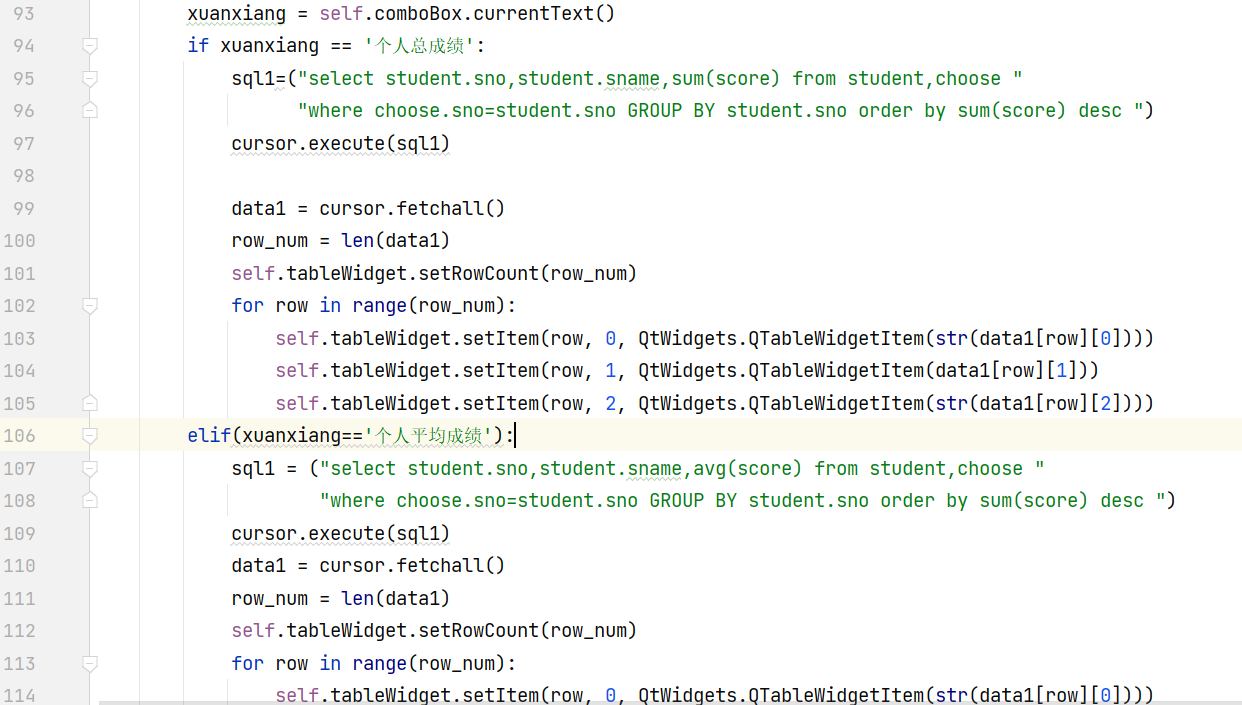
4.2.5学生信息增加 这部分的实现是获取用户输入的信息来更新数据库，部分代码如下：



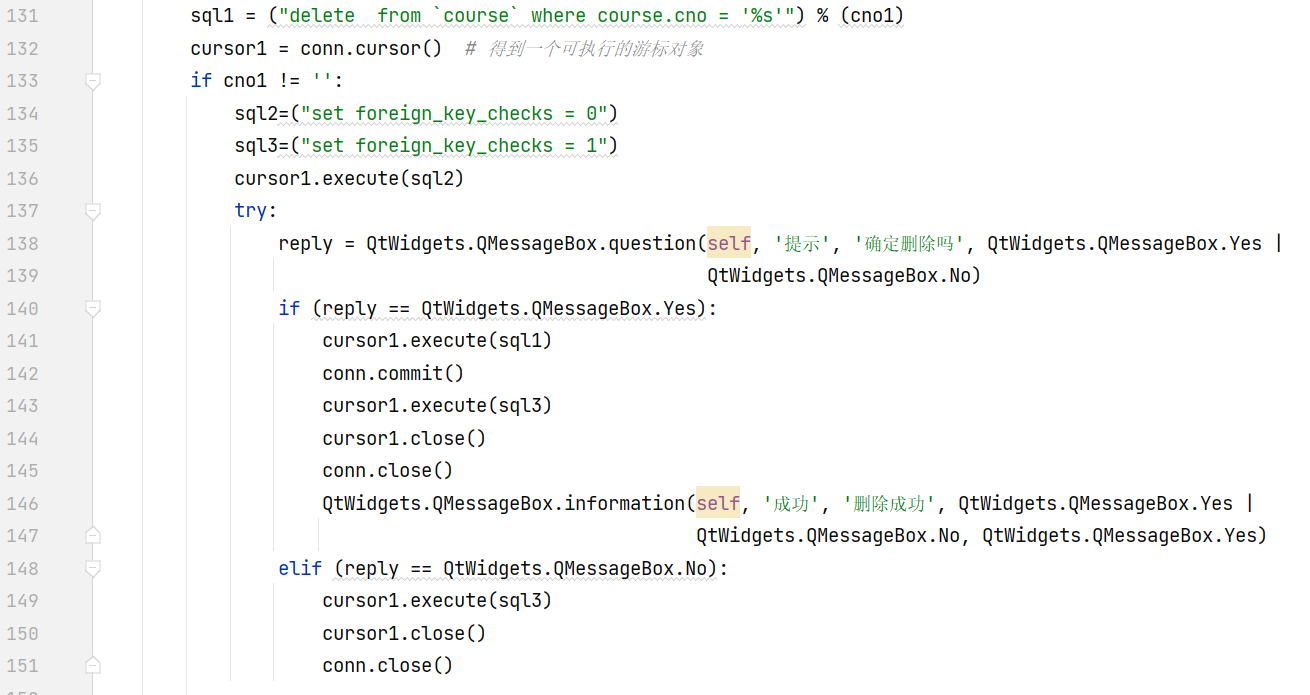
4.2.6班级之间的排名 班级之间的排名方式分为按班级总分以及班级平均分排名，使用者可以通过不同的排名方式查询班级排名，这部分可以利用sql语言实现排序，不需要使用python语言排序，部分代码如下：



4.2.7学生个人成绩排名 将学生成绩按不同条件排名，如总成绩，单科成绩，平均成绩；排名功能与4.2.6类似，代码部分与4.2.6基本相同，SQL语句做出修改，故只附上部分代码：



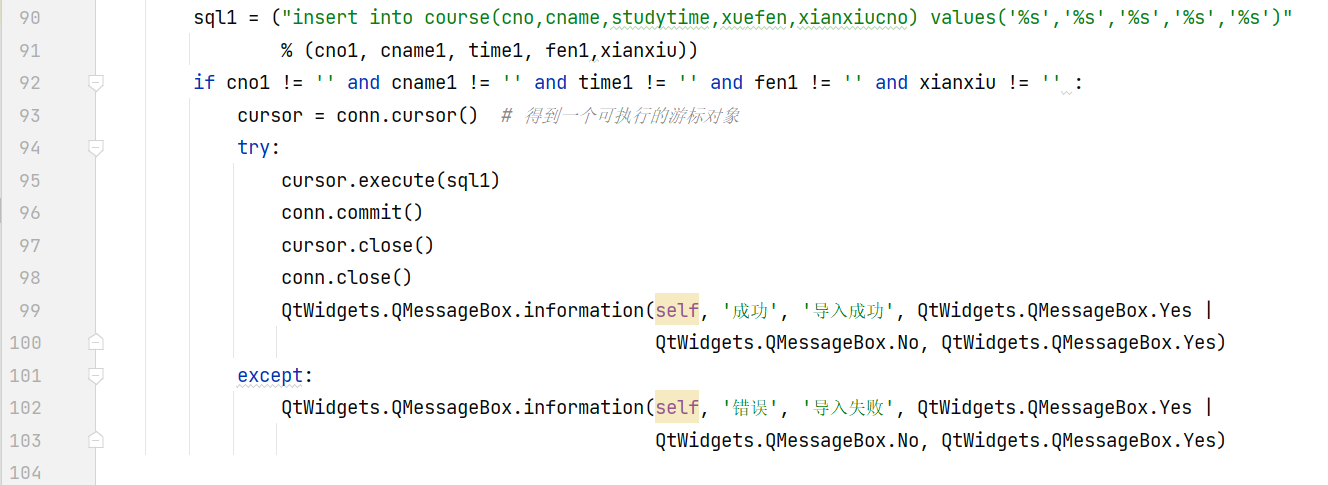
4.2.8 课程信息删除，这部分功能仍然使用查询功能与删除功能，与4.2.3类似，查询和删除对象由学生信息转换成课程信息，SQL语句做出修改，同时也需要将外键约束关闭，待操作结束后再打开，部分代码如下；



4.2.9 课程信息修改，这部分类似4.2.4，同样用到了查询功能和修改功能，重复部分不再赘述，部分代码如下：



4.2.10课程信息增加，功能类似于学生信息增加，部分代码如下：

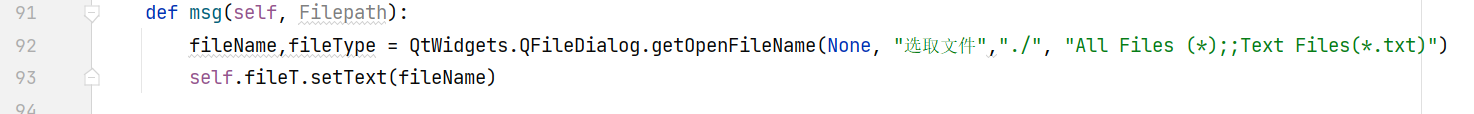


4.2.11奖学金统计 这部分功能是自动生成的，通过sql语言查询数据库中的学生成绩，进行排序，选取成绩前20的学生显示在表格中，并且奖学金的等级也不同，部分代码如下图所示：

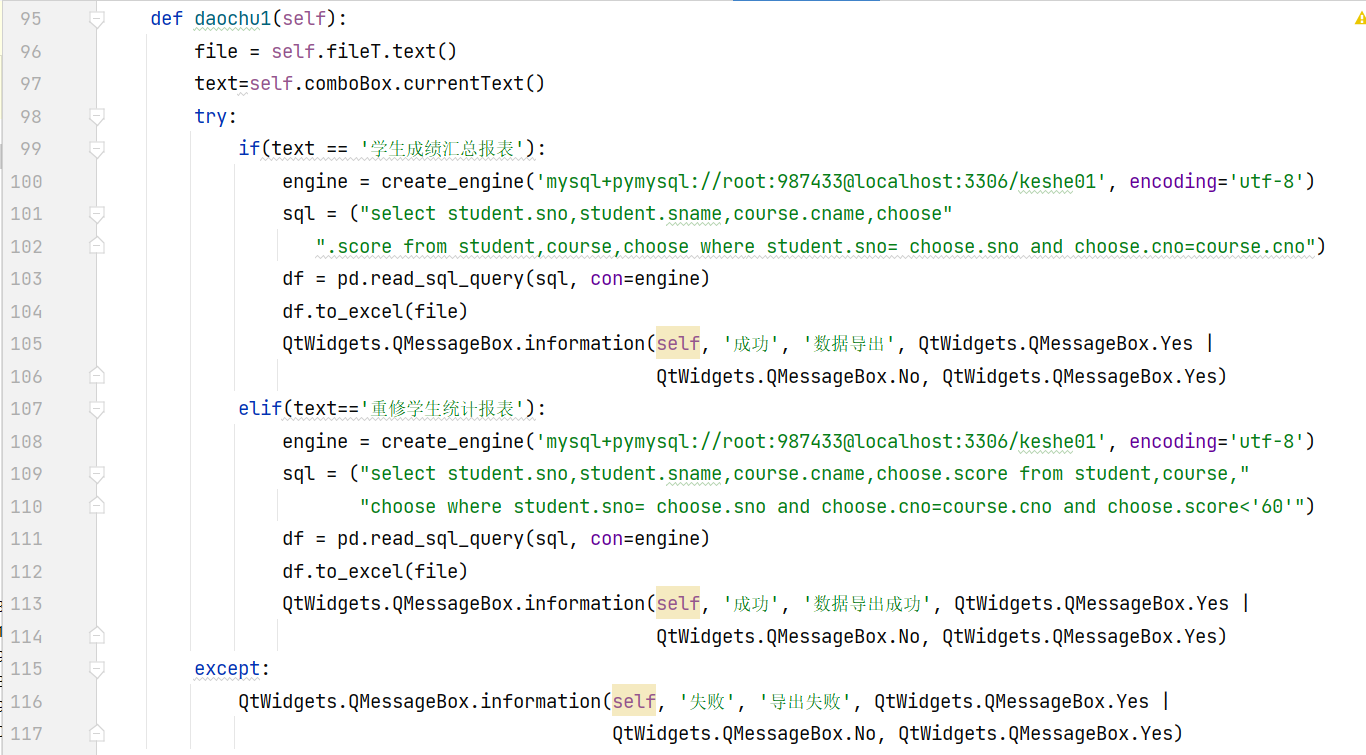


4.2.12报表输出与数据维护 这个功能将报表统计与数据维护统一起来，用户可以选择系统文件的绝对路径，将学生成绩汇总报表等报表导出到该文件路径中，注意选择文件格式，而数据维护也是选择系统文件的绝对路径，将数据库转储在该文件中，数据恢复同理，将该文件导入到数据库中，实现数据恢复，同样需要注意文件格式。部分代码如下：

下图为选择文件路径实现：



下图为学生信息报表导出，file为保存的文件路径

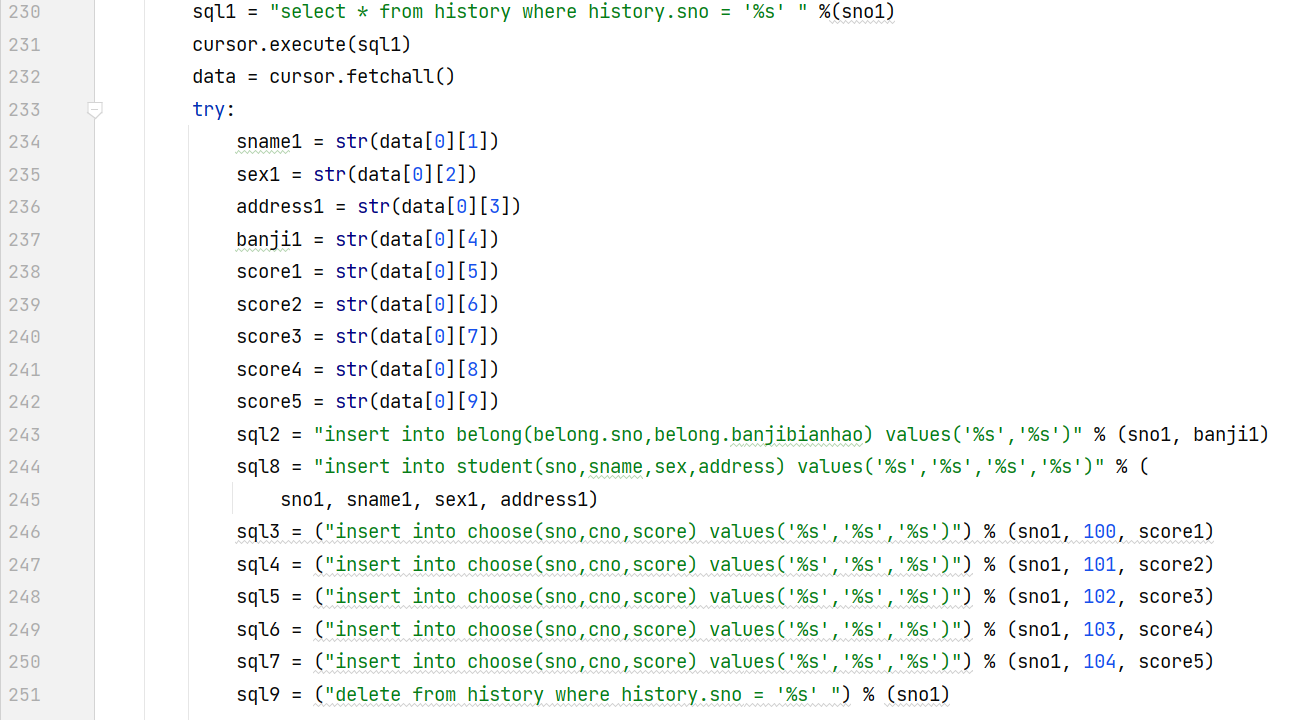


使用create\_engine连接数据库，pd.read\_sql\_query读取信息

下图为数据备份与数据恢复：



4.2.13历史学生 该部分实现删除学生的恢复以及查询，查询功能同上面的查询实现类似，不再赘述，恢复功能则是通过读取历史学生数据表的数据，然后将其数据插入到现有的学生信息表中，同时也会将这个学生的数据从历史学生数据表中删除，部分代码如下图：



4.2.14 信息的插入 数据库应该包含成百上千条数据，如果自己手动插入花费大量时间，因此我们通过读取表格信息将数据插入到数据库，具体代码如下：



**5.系统测试**

1.数据库模式测试

我们小组内部在开发阶段进行了基本表数据结构测试和数据库完整性测试，对模块所有独立的执行路径测试了几遍，在进行基本表数据结构测试并未发现问题，但是在完整性测试时发现了不少错误，如约束设置不合理,导致出现了学号为0，成绩大于100分的情况出现，经过不断地对数据库进行改进，这些错误也被解决。

2.数据库功能测试

通过测试用例运行数据库，测试数据库的功能是否正确运行，我们通过可视化软件navicat查询数据库，将修改后的数据库与navicat中的数据对比，反查功能是否正常运行，测试基本完成，功能无错误

3．数据库性能测试

我们对数据库进行了压力测试，向数据库插入了4000条数据，系统可以正常运行，但在进行班级总分排序时出现卡顿，甚至出现未响应，原因初步排查是由于表的结构不合理，一个人成绩需要占据五行，数据过大，mysql计算数据需要花费太多时间，但这个排序方式根据需求分析的结论，我们觉得是可以取消的，原因是班级总分排序不合理，每个班级人数不同，不能以此作为班级是否优秀的依据。

