数据库实验一之 SQL 练习

设有如下关系模式：
Student(SNO, NAME, GENDER, BIRTHDAY, DEPART)
SNO 为学生学号，DEPART 为系别
Teacher(TNO,NAME,GENDER,BIRTHDAY,POSITION,DEPART)
TNO 为教师工号，POSITION 为职称，DEPART 为系别。
Course(CNO, NAME, TNO)
CNO 为课程编号，TNO 为教师工号。
Score(SNO, CNO, DEGREE)
SNO 为学生学号，CNO 为课程编号，DEGREE 为成绩。
PS:加下划线"___"的表示该字段为主键。

任务一：根据所给 Student.csv、Teacher.csv、Course.csv、Score.csv 表中的数据信息，在数据库中创建对应的关系表并将数据录入到数据库中。可能涉及到的数据类型：varchar, char, int, float ,datetime。（**也需要截图展示!**）

任务二：**依顺序**写出实现以下各题功能的 SQL 语句（部分题目的结果受前面题目影响）：

注意：严禁抄袭!!!

修改基本表：

1、在学生表 student 中增加一个新的属性列 AGE(年龄)，类型为 int。

2、计算每个学生的年龄(AGE)（简单用 2023 减去出生年份即可）。注意，此操作可能需要关闭安全更新模式；提示，可使用 MySQL 的 YEAR 函数。

3、为每个学生的年龄加 2。

4、将 AGE（年龄）的数据类型由 int 改为 char。

5、删除属性列 AGE。

6、创建一个教师课程数量表：teacher_course(TNO,NUM_COURSE)，两个属性分别表示授课教师工号，课程数量，类型自定义(注意，这里 TNO 还不是主键)。

7、为表 teacher_course 添加主键(TNO)。

8、用一条语句，结合表 course 记录，为表 teacher 中所有教师，在表 teacher_course 添加对应记录（若是表 course 中未出现的教师，则课程数量记为 NULL）。

9、删除表 teacher_course 中含有 NULL 的记录。

10、删除表 teacher_course。

11、在学生表 student 、成绩表 score 中分别插入一些数据，数据如下(注意: 如果与原有数据冲突，比如学号重复，请修改一下自己的信息保证能够插入):

SNO	NAME	GENDER	BIRTHDAY	DEPART
{你的学号}	{你的名字（首字母缩写）}	{你的性别（male,female）}	{你的出生日期}	229（统一采用 229）
{你的好友的学号}	{你的好友的名字（首字母缩写）}	{你的好友的性别（male,female）}	{你的好友的出生日期}	10-14 之间或者 229 即可
{你的好友的学号}	{你的好友的名字（首字母缩写）}	{你的好友的性别（male,female）}	{你的好友的出生日期}	10-14 之间或者 229 即可

SNO	CNO	DEGREE
{你的学号}	20230402	95-99 之间
{你的学号}	20230410	95-99 之间
{你的学号}	20230412	95-99 之间

请保证选择的三门课成绩**不一样**。

12、在 score 表中删除你所选的课程中成绩最低的一门课程的记录（可能会用到子查询）。

索引：

13、用 create 语句在 course 表的名称 NAME 上建立普通索引 NAME_INDEX。

14、用 create 语句在 teacher 表的工号 TNO 上建立唯一索引 TNO_INDEX。

15、用 create 语句在 score 表上的学号 SNO、成绩 DEGREE 上建立复合索引 RECORD_INDEX, 要求学号为降序，学号相同时成绩为升序。

16、用一条语句查询表 score 的索引。

17、删除 teacher 表字段 TNO 上的索引 TNO_INDEX

查询：

注：以下每道题限制用一条 SQL 语句完成

18、查询和你属于同一个系的学生学号和姓名(**包括**你本人)。

19、查询和你属于同一个系的学生学号和姓名(**不包括**你本人)。

20、查询和你的某个好友属于同一个系的学生学号和姓名（11 题插入的某个好友)。

21、查询和你的两个好友都不在一个系的学生学号和姓名（11 题插入的两个好友)。

22、查询教过你的所有老师的工号和姓名。

23、查询 11 系和 229 系教师的总人数。

24、查询选修 DB_Design 课程且成绩在 89 分以上（包括 89）的学生的学号、姓名和分数。

25、查询选修过“ZDH”老师课程的学生学号和姓名（去掉重复行)。

26、查询选过某课程的学生学号和分数，并按分数降序展示。(某课程是指 course 表中的某一课程名 NAME，你自行选择)。

27、查询每门课的平均成绩，其中每行包含课程号、课程名和平均成绩（包括平均成绩为 NULL，即该课没有成绩)。

28、查询每门课程的最高分和最低分，并计算其分数差。其中每行包含课程号、课程名和最高分、最低分和分数差。(课程无成绩的不用包括)。

29、查询所教过的课程中有学生考试成绩低于 72 分的教师的工号和姓名（去掉重复行)。

30、查询选修了 2 门课程及以上的学生的学号、姓名。

31、查询 student 表中各个学生姓名与相应的平均成绩(没有选课的学生平均成绩为 NULL)。

32、查询每个系的学生人数和每个系的平均分，其中每行包含系号、系的人数和平均成绩。这里平均成绩是指每个学生的所有课程的平均成绩计算后，与同一个系的其他同学再次计算平均值。

33、查询所有未选修 Data_Mining 课程的学生姓名（去掉重复行)。

34、查询各个课程的课程名及选该课的学生的平均年龄。(包括没有人选的课)

35、查询选修了课程名中包含”Computer”课程的学生的学号和姓名。

36、查询成绩比该课程平均成绩高 12 分以上的同学的成绩表，即包含 SNO、CNO、DEGREE。

视图：

37、建立女学生的学生视图（db_female_student)，属性与 student 表一样，并要求对该视图进行修改和插入操作时仍需保证该视图只有女学生。

38、将女学生视图（db_female_student）中学号为“PB210000016”的学生姓名改为{你的姓名（英文首字母）}。

39、在女学生视图（db_female_student）中找出年龄小于 21 岁的学生，包含 SNO、NAME。

40、向 student 表中插入一名“学号 SA210110021，姓名 QXY，性别女，生日 1997/7/27，12 系”的学生。然后查询视图 db_female_student 的所有学生，验证其是否更新。

41、向视图 db_female_student 中插入一名“学号 SA210110023，姓名 DPC，性别男，生日 1997/4/27，11 系”的学生，观察到了什么现象？

42、删除视图 db_female_student。

触发器：

43、创建关系表：teacher_salary(TNO, SAL)，其中 TNO 是教师工号（主键），SAL 是教师工资（类型 float）。

44、定义一个 BEFORE 行级触发器，为关系表 teacher_salary 定义完整性规则：“表中出现的工号必须也出现在 teacher 表中，否则报错”。注：该规则实际上就是外键约束；MySQL 中可使用 SIGNAL 抛出错误；需要为 INSERT 和 UPDATE 分别定义触发器。请展示出成功创建触发器和测试抛出错误信息的截图。

45、定义一个 BEFORE 行级触发器，为关系表 teacher_salary 定义完整性规则：“Instructor/Associate Professor/Professor 的工资不能低于 4000/5000/6000，如果低于，则改为 4000/5000/6000”。注：需要为 INSERT 和 UPDATE 分别定义触发器。并检验触发器是否工作：为 teacher_salary 构造 INSERT 和 UPDATE 语句并激活所定义过的触发器，将过程截图展示。

46、删除刚刚创建的所有触发器。

空值：

47、将 score 表中的 Data_Mining 课程成绩设为空值，然后在 score 表查询学生学号和分数，并按分数升序展示。观察 NULL 在 MySQL 中的大小是怎样的？

开放题：

48、49、50 请自己设计三个题目并实现。