SQL 经典 50 题-tidyverse 版

张敬信

2022-02-23

关于作者：

- 张敬信，哈尔滨商业大学，数学与应用数学，副教授
- 热爱学习，热爱编程，热爱 R 语言
- 我正在用最新 R 技术写一本《R 语言编程—基于 tidyverse》的书，欢迎您的阅读品鉴！

  该书的 知乎交流平台, 欢迎您的留言讨论！

  该书的 QQ 读者群：875664831, 交流、答疑，欢迎您的加入！

MySQL/SQL 经典 50 题在互联网上广为流传，其中一个比较精美的版本是：来自 **Python 数据之道**的《**超经典 MySQL 练习 50 题**》。

R 语言的 tidyverse 包，涵盖整个数据科学流程，也包括 SQL 数据库功能，数据操作整洁、优雅，读它的代码就像读文字叙述一样。

阅读本文档内容，需要了解一点最基本的管道操作：

管道符号 `%>%` 的作用是将前面的数据自动往前输送，依次用于每步操作，从此每步代码不用再管数据，只关心每步如何操作数据即可。

所以，我[1][2] 推出一个 tidyverse 版本，以推广 tidyverse 和最新的 R 语言技术。

想要深入学习 tidyverse 和最新的 R 语言技术，请移步 《R 语言编程—基于 tidyverse》 学习交流！

该版本是源于我的 R 语言新书读者群（QQ 群：875664831）里发起的一轮数据操作的练习活动，部分题目也收录了群友” 你为长安”、“PORPOIS”、“偏居一隅” 等的解法，一并表示感谢！

[1]我的 Github: https://github.com/zhjx19

[2]我的知乎: https://www.zhihu.com/people/huc_zhangjingxin

- 加载包

```
library(tidyverse)
library(lubridate)
```

创建表

- 按列创建表用 `tibble()`, 按行录入式地创建表用 `tribble()`

```
student = tribble(
  ~学号, ~姓名, ~生日, ~性别,
  '01', '赵雷', '1990-01-01', '男',
  '02', '钱电', '1990-12-21', '男',
  '03', '孙风', '1990-05-20', '男',
  '04', '李云', '1990-08-06', '男',
  '05', '周梅', '1991-12-01', '女',
  '06', '吴兰', '1992-03-01', '女',
  '07', '郑竹', '1989-07-01', '女',
  '08', '王菊', '1990-01-20', '女'
)
student
```

```
## # A tibble: 8 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
## 5 05    周梅  1991-12-01 女
## 6 06    吴兰  1992-03-01 女
## # ... with 2 more rows
```

```
score = tribble(
  ~学号, ~课程编号, ~成绩,
  '01', '01', 80,
  '01', '02', 90,
  '01', '03', 99,
  '02', '01', 70,
  '02', '02', 60,
  '02', '03', 80,
  '03', '01', 80,
```

```
  '03', '02', 80,
  '03', '03', 80,
  '04', '01', 50,
  '04', '02', 30,
  '04', '03', 20,
  '05', '01', 76,
  '05', '02', 87,
  '06', '01', 31,
  '06', '03', 34,
  '07', '02', 89,
  '07', '03', 98
)
score
```

```
## # A tibble: 18 x 3
##   学号  课程编号  成绩
##   <chr> <chr>    <dbl>
## 1 01    01          80
## 2 01    02          90
## 3 01    03          99
## 4 02    01          70
## 5 02    02          60
## 6 02    03          80
## # ... with 12 more rows
```

```
course = tribble(
  ~课程编号, ~课程名称, ~教师编号,
  '01', '语文', '02',
  '02', '数学', '01',
  '03', '英语', '03'
)
course
```

```
## # A tibble: 3 x 3
##   课程编号 课程名称 教师编号
##   <chr>    <chr>    <chr>
## 1 01       语文     02
## 2 02       数学     01
## 3 03       英语     03
```

```
teacher = tribble(
  ~教师编号, ~教师姓名,
  '01', '张三',
  '02', '李四',
  '03', '王五'
)
teacher
```

```
## # A tibble: 3 x 2
##   教师编号 教师姓名
##   <chr>    <chr>
## 1 01       张三
## 2 02       李四
## 3 03       王五
```

- 保存上述 4 个数据表，以方便后续使用

```
save(course, score, student, teacher,
     file = "SQL50datas.rda")
load("SQL50datas.rda")          # 载入数据
```

1. 查询”01” 课程比”02” 课程成绩高的学生的信息及课程分数

- 先长变宽重塑为宽表，然后根据条件筛选行，再连接学生信息

```
# 为了方便理解，先看一看这个宽表
score1 = score %>%
  pivot_wider(names_from = 课程编号, values_from = 成绩,
              names_prefix = " 课程")
score1
```

```
## # A tibble: 7 x 4
##   学号  课程01 课程02 课程03
##   <chr>  <dbl>  <dbl>  <dbl>
## 1 01        80     90     99
## 2 02        70     60     80
## 3 03        80     80     80
## 4 04        50     30     20
## 5 05        76     87     NA
## 6 06        31     NA     34
## # ... with 1 more row
```

```
score1 %>%
  filter(课程 01 > 课程 02) %>%
  left_join(student, by = " 学号")
```

```
## # A tibble: 2 x 7
##   学号  课程01 课程02 课程03 姓名  生日       性别
##   <chr>  <dbl>  <dbl>  <dbl> <chr> <chr>      <chr>
## 1 02        70     60     80 钱电  1990-12-21 男
## 2 04        50     30     20 李云  1990-08-06 男
```

2. 查询"01" 课程比"02" 课程成绩低的学生的信息及课程分数

- 直接使用新创建的宽表

```
score1 %>%
  filter(课程 01 < 课程 02) %>%
  left_join(student, by = " 学号")
```

```
## # A tibble: 2 x 7
##   学号  课程01 课程02 课程03 姓名  生日       性别
##   <chr>  <dbl>  <dbl>  <dbl> <chr> <chr>      <chr>
## 1 01        80     90     99 赵雷  1990-01-01 男
## 2 05        76     87     NA 周梅  1991-12-01 女
```

3. 查询平均成绩大于等于 60 分的学生学号、姓名和平均成绩

- 先分组汇总计算平均成绩，然后根据条件筛选行，再连接学生信息，最后选择列

```
score %>%
  group_by(学号) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  filter(平均成绩 >= 60) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 平均成绩)
```

```
## # A tibble: 5 x 3
##   学号  姓名  平均成绩
##   <chr> <chr>    <dbl>
## 1 01    赵雷      89.7
## 2 02    钱电      70
## 3 03    孙风      80
## 4 05    周梅      81.5
## 5 07    郑竹      93.5
```

附加题：总分超过 200 分的学生学号、姓名、总分

- 思路相同，只是从汇总平均分改成汇总总分

```
score %>%
  group_by(学号) %>%
  summarise(总分 = sum(成绩)) %>%
  filter(总分 > 200) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 总分)
```

```
## # A tibble: 3 x 3
##   学号  姓名   总分
##   <chr> <chr> <dbl>
## 1 01    赵雷    269
## 2 02    钱电    210
## 3 03    孙风    240
```

4. 查询平均成绩小于 60 分的学生的学号和姓名和平均成绩（包括有成绩的和无成绩的）

- 数据中是有一名学生没有任何考试成绩的，如果不考虑该生，即只考虑有成绩的学生
- 先以成绩表作为左表连接学生信息，然后分组汇总计算平均成绩，再根据条件筛选行

```
score %>%
  left_join(student, by = " 学号") %>%
  group_by(学号, 姓名) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  filter(平均成绩 < 60)
```

```
## # A tibble: 2 x 3
## # Groups:   学号 [2]
##   学号  姓名  平均成绩
##   <chr> <chr>    <dbl>
## 1 04    李云      33.3
## 2 06    吴兰      32.5
```

- 若考虑所有学生，无论有没有成绩，没有成绩做 0 分处理
- 以学生表作为左表连接成绩表，将缺失成绩插补为 0 分，再做上述事情

```
student %>%
  left_join(score, by = " 学号") %>%
  replace_na(list(成绩 = 0)) %>%
  group_by(学号, 姓名) %>%
  summarise(平均成绩 = mean(成绩)) %>%
```

```
  filter(平均成绩 < 60)
```

```
## # A tibble: 3 x 3
## # Groups:   学号 [3]
##   学号  姓名  平均成绩
##   <chr> <chr>    <dbl>
## 1 04    李云      33.3
## 2 06    吴兰      32.5
## 3 08    王菊       0
```

5. 查询所有学生的学号、姓名、选课总数、所有课程的总成绩

- 同上面一样，改用另一种方式：mutate() 修改列的方式，将缺失成绩插补为 0 分
- 再做分组汇总，函数 n_distinct() 是对唯一值计数，因为没有成绩学生的课程是缺失，计算时需忽略缺失值

```
student %>%
  left_join(score, by = " 学号") %>%
  mutate(成绩 = replace_na(成绩, 0)) %>%
  group_by(学号, 姓名) %>%
  summarise(选课总数 = n_distinct(课程编号, na.rm = TRUE),
            总成绩 = sum(成绩))
```

```
## # A tibble: 8 x 4
## # Groups:   学号 [8]
##   学号  姓名  选课总数 总成绩
##   <chr> <chr>    <int>  <dbl>
## 1 01    赵雷         3    269
## 2 02    钱电         3    210
## 3 03    孙风         3    240
## 4 04    李云         3    100
## 5 05    周梅         2    163
## 6 06    吴兰         2     65
## # ... with 2 more rows
```

6. 查询”李”姓老师的数量

- 先计算新列，str_detect() 检测字符串匹配返回 TRUE/FALSE，"^ 李" 是正则表达式，匹配以”李”开头
- 对逻辑值列求和就是计数

```
teacher %>%
  mutate(idx = str_detect(教师姓名, "^ 李")) %>%
  summarise(n = sum(idx))
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     1
```

- 或者，直接用 count() 做分组计数，需提供分组变量或分组条件表示的分组变量，本例是后者

```
teacher %>%
  count(是否李姓 = str_detect(教师姓名, "^ 李"))
```

```
## # A tibble: 2 x 2
##   是否李姓     n
##   <lgl>    <int>
## 1 FALSE        2
## 2 TRUE         1
```

7. 查询学过张三老师教授课程的学生信息

- 先以教师表作为左表连接课程表，根据条件筛选行，得到张三老师教授课程
- 然后根据上步选出的课程连接成绩表，得到学过相应课程的学号
- 再用上步选出的学号去**筛选**学生表得到学生信息，半连接 semi_join() 正好适合：根据与第 2 个表匹配筛选第 1 个表，. 代表上一步通过管道输送过来的数据，用做第 2 个表
- 管道输送的数据，默认是用于函数的第一个参数，并默认省略不写，若想用于非第一个参数时，必须用 . 来代替该数据

```
teacher %>%
  left_join(course, by = " 教师编号") %>%
  filter(教师姓名 == " 张三") %>%
  left_join(score, by = " 课程编号") %>%
  semi_join(student, ., by = " 学号")
```

```
## # A tibble: 6 x 4
##   学号  姓名  生日        性别
##   <chr> <chr> <chr>       <chr>
## 1 01    赵雷  1990-01-01  男
## 2 02    钱电  1990-12-21  男
## 3 03    孙风  1990-05-20  男
## 4 04    李云  1990-08-06  男
## 5 05    周梅  1991-12-01  女
```

```
## 6 07    郑竹   1989-07-01 女
```

8. 找出没有学过张三老师教授课程的学生信息

- 与上一题相似，**筛选**换成**反选**，只需要将 semi_join() 换成反连接 anti_join()

```
teacher %>%
  left_join(course, by = " 教师编号") %>%
  filter(教师姓名 == " 张三") %>%
  left_join(score, by = " 课程编号") %>%
  anti_join(student, ., by = " 学号")
```

```
## # A tibble: 2 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 06    吴兰  1992-03-01 女
## 2 08    王菊  1990-01-20 女
```

9. 查询学过"01" 和"02" 课程的学生信息

- 成绩宽表 score1 课程列不是 NA 就是学过，作为条件筛选行
- 再用于半连接筛选学生表，得到学生信息

```
score1 %>%
  filter(!is.na(课程 01), !is.na(课程 02)) %>%
  semi_join(student, ., by = " 学号")
```

```
## # A tibble: 5 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
## 5 05    周梅  1991-12-01 女
```

- **"你为长安" 的解法：**先分别选出两个课程的学号，再依次与学生表做半连接筛选学生信息

```
class1 = score %>%
  filter(课程编号 == "01")
class2 = score %>%
  filter(课程编号 == "02")
class1 %>%
  semi_join(class2, by = " 学号") %>%
```

```
  semi_join(student, ., by = " 学号")
```

10. 查询学过"01"课程，但没有学过"02"课程的学生信息

- 道理同上题 (略)

```
score1 %>%
  filter(!is.na(课程 01), is.na(课程 02)) %>%
  semi_join(student, ., by = " 学号")
```

```
## # A tibble: 1 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 06    吴兰  1992-03-01 女
```

- "你为长安"的解法

```
class1 = score %>%
  filter(课程编号 == "01")
class2 = score %>%
  filter(课程编号 == "02")
class1 %>%
  anti_join(class2, by = " 学号") %>%
  semi_join(student, ., by = " 学号")
```

11. 查询没有学完全部课程的学生信息

- 假设成绩表中出现的课程就是全部课程，直接使用成绩宽表
- 左连接学生表，获得学生信息
- 没有学完全部课程，即 3 个课程列存在 NA，根据多列的值构造判断条件做筛选行，适合用 filter + if_any/if_all 实现，多列存在值满足条件用 if_any(选择列, 判断条件)

```
score1 %>%
  right_join(student, by = " 学号") %>%
  filter(if_any(2:4, is.na)) %>%
  select(-(2:4))
```

```
## # A tibble: 4 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 05    周梅  1991-12-01 女
## 2 06    吴兰  1992-03-01 女
## 3 07    郑竹  1989-07-01 女
## 4 08    王菊  1990-01-20 女
```

- 更稳妥的做法是从课程表出发，用下面的结果表代替 score1:

```
course %>%
  left_join(score, by = " 课程编号") %>%
  select(-c(课程编号, 教师编号)) %>%
  pivot_wider(names_from = 课程名称, values_from = 成绩)
```

- **“PORPOIS” 的解法：**成绩宽表剔除 3 个课程列包含 NA 的行，就是选择全部课程的学生，用其学号反选 (反连接) 学生表即可

```
score1 %>%
  drop_na(课程 01, 课程 02, 课程 03) %>%
  anti_join(student, ., by = " 学号")
```

- **“偏居一隅” 的解法 (很巧妙地把课程表用进来)：**成绩表根据学号分组，n() 即每个学生学完的课程数，若等于总课程数则筛选出来，再反连接学生表

```
score %>%
  group_by(学号) %>%
  filter(n() == nrow(course)) %>%
  anti_join(student, ., by = " 学号")
```

12. 查询至少有一门课与学生”01” 所学课程相同的学生信息

- 成绩表筛选学生”01” 的行，其中的课程编号就是该生所学课程
- 用来与成绩表做半连接，筛选成绩表中这些课程的行，得到学过这些课程的学生学号
- 再与学生表做半连接，筛选出想要的学生信息

```
score %>%
  filter(学号 == "01") %>%
  semi_join(score, ., by = " 课程编号") %>%
  semi_join(student, ., by = " 学号")
```

```
## # A tibble: 7 x 4
##   学号  姓名  生日        性别
##   <chr> <chr> <chr>       <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
## 5 05    周梅  1991-12-01 女
## 6 06    吴兰  1992-03-01 女
## # ... with 1 more row
```

13. 查询与学生”01” 学习的课程完全相同的学生信息

- 先从成绩表筛选学生”01” 的行，准备好该生的所有课程编号
- 用 group_nest() 对成绩表按学号做分组嵌套，这样的好处是，每个学生的数据占一行，其所有课程及成绩信息打包成一个对象 (数据框)。为了方便理解，打断管道操作先看一下它：

```
c01 = score %>%
  filter(学号 == "01")

score_nest = score %>%
  group_nest(学号)
score_nest
```

```
## # A tibble: 7 x 2
##   学号                data
##   <chr> <list<tibble[,2]>>
## 1 01               [3 x 2]
## 2 02               [3 x 2]
## 3 03               [3 x 2]
## 4 04               [3 x 2]
## 5 05               [2 x 2]
## 6 06               [2 x 2]
## # ... with 1 more row
```

```
score_nest$data[[1]]
```

```
## # A tibble: 3 x 2
##   课程编号  成绩
##   <chr>    <dbl>
## 1 01          80
## 2 02          90
## 3 03          99
```

- 从嵌套成绩表继续，要筛选满足条件的学号，条件是其所有课程编号与学生”01” 的相同，即两个课程编号集合相等
- 筛选行用 filter(), 构造一个逻辑向量作为筛选条件，该逻辑向量由判断课程编号集合是否相等得到
- 这相当于依次从 data 列中取出其课程编号集合，与学生”01” 的课程编号集合，做是否集合相等判断
- 这种循环迭代操作，适合用 purrr 包的 map 系列来做，返回类型是逻辑值，所以用带后缀的 map_lgl()
- 其第 1 个参数 data 表明是在 data 列上依次迭代，第 2 个参数是要应用的函数 (判断集合相等)，这里是用 purrr-风格公式 (匿名函数) 写法，形参 .x 代指一元函数的自变量
- 在 map 机制下，该函数将依次应用到 data 列的每个元素上，完成咱们想要的判断
- 这样筛选出的学号，再用半连接筛选学生表，得到最终结果

```
score_nest %>%
  filter(map_lgl(data, ~ setequal(.x$课程编号, c01$课程编号))) %>%
  semi_join(student, ., by = " 学号")
```

```
## # A tibble: 4 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
```

上述做法代码不复杂 (去掉中间变量把管道接一起)，但是很抽象难以理解。受 **“偏居一隅”** 启发，与其放一起作为集合比较集合相等，何不拼接为一个字符串，比较字符串？

- 对成绩表按学号分组，将同一学号所选课程，合并为一个字符串。为了不受先后顺序的影响，应该先用 `arrange()` 对学号排个序
- 然后筛选出所选课程汇编字符串与学生”01” 相同的学号
- 再与学生表做半连接，筛选学生信息

```
score %>%
  arrange(学号) %>%
  group_by(学号) %>%
  summarise(课程汇编 = str_c(课程编号, collapse = ",")) %>%
  filter(课程汇编 == .$课程汇编 [学号=="01"]) %>%
  semi_join(student, ., by = " 学号")
```

注:本题可以说所有题目中最难的一道,虽然很啰嗦但穿插讲解到很多语法,还是很有意义的。另外,`group_by + summarise(str_c)` 是 `separate_rows()` 的逆操作。

14. 同 8. (略)

15. 查询两门及以上不及格课程的学生学号，姓名及其平均成绩

- 从成绩宽表开始，用 `filter()` 根据条件筛选行，条件是两门及以上课程不及格
- 三门课程占据 3 列，先判断是否不及格，再对逻辑值加和，就是不及格的门数 (要注意忽略 `NA`)，再判断是否 2
- 每一行是一个学生，相当于依次对每一行做出判断，这叫做数据框逐行迭代，`purrr` 包中的 `pmap` 系列就是做这个的，想要返回逻辑值作为筛选条件，所以用带后缀的 `pmap_lgl()`
- 其第 1 个参数就是三门课程列构成的数据框，第 2 个参数是要应用的函数 (判断条件)，其形参 `...` 代表使用该行所有的列值，注意需要套一个 `c()` 打包成一个对象
- 上述筛选就选出了满足条件的学生，接着是计算平均成绩，计算新列用 `mutate()`, 基本格式是 新列名 = 计算表达式

- 仍是逐行迭代计算平均值，注意忽略 NA, 仍用 pmap 系列实现，区别是这次返回的是数值，所以用带后缀的 pmap_dbl()
- 最后，再连接学生信息进来，若不想带着多余列，用 select() 选择想要的列

```
score1 %>%
  filter(pmap_lgl(.[-1], ~ sum(c(...) < 60, na.rm = TRUE) >= 2)) %>%
  mutate(平均成绩 = pmap_dbl(.[-1], ~ mean(c(...), na.rm = TRUE))) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 平均成绩)
```

```
## # A tibble: 2 x 3
##   学号  姓名  平均成绩
##   <chr> <chr>    <dbl>
## 1 04    李云      33.3
## 2 06    吴兰      32.5
```

16. 检索"01" 课程分数小于 60，按分数降序排列的学生信息

- 用 filter() 筛选行，其实是两个筛选条件
- 把学生信息连接进来，用 select() 选择列：删除课程编号列
- 最后用 arrange() 根据成绩列排序，默认是增序，要降序加-或套一个 desc()

```
score %>%
  filter(课程编号 == "01", 成绩 < 60) %>%
  left_join(student, by = " 学号") %>%
  select(-课程编号) %>%
  arrange(-成绩)
```

```
## # A tibble: 2 x 5
##   学号   成绩 姓名  生日       性别
##   <chr> <dbl> <chr> <chr>      <chr>
## 1 04       50 李云  1990-08-06 男
## 2 06       31 吴兰  1992-03-01 女
```

17. 按平均成绩从高到低显示所有学生的所有课程的成绩以及平均成绩

- 对成绩表按学号做分组汇总，计算平均成绩，再对平均成绩按降序排序

```
# 长表
score %>%
  group_by(学号) %>%
  mutate(平均成绩 = mean(成绩)) %>%
  arrange(-平均成绩)
```

```
## # A tibble: 18 x 4
## # Groups:   学号 [7]
##   学号  课程编号  成绩 平均成绩
##   <chr> <chr>    <dbl>    <dbl>
## 1 07    02          89     93.5
## 2 07    03          98     93.5
## 3 01    01          80     89.7
## 4 01    02          90     89.7
## 5 01    03          99     89.7
## 6 05    01          76     81.5
## # ... with 12 more rows
```

- 上面是按长表实现，若要按成绩宽表实现，同 15. 做法逐行迭代计算平均成绩再排序

```
score1 %>%
  mutate(平均成绩 = pmap_dbl(.[-1], ~ mean(c(...), na.rm = TRUE))) %>%
  arrange(-平均成绩)
```

```
## # A tibble: 7 x 5
##   学号  课程01 课程02 课程03 平均成绩
##   <chr>  <dbl>  <dbl>  <dbl>    <dbl>
## 1 07        NA     89     98     93.5
## 2 01        80     90     99     89.7
## 3 05        76     87     NA     81.5
## 4 03        80     80     80     80
## 5 02        70     60     80     70
## 6 04        50     30     20     33.3
## # ... with 1 more row
```

18. 查询各科成绩最高分、最低分和平均分，以如下形式显示：

课程编号，课程名称，最高分，最低分，平均分，及格率，中等率，优良率，优秀率；

注：及格：>=60，中等为：70-80，优良为：80-90，优秀为：>=90

- 分组汇总，这里用到一个小技巧是：**对是否满足某条件的逻辑向量取平均，就是该条件的占比**
- `relocate()` 函数用来调整列序

```
score %>%
  group_by(课程编号) %>%
  summarise(最高分 = max(成绩),
            最低分 = min(成绩),
            平均分 = mean(成绩),
            及格率 = mean(成绩 >= 60),
```

```
            中等率 = mean(成绩 >= 70 & 成绩 < 80),
            优秀率 = mean(成绩 >= 90)) %>%
  left_join(course, by = " 课程编号") %>%
  relocate(课程名称, .after = 课程编号) %>%
  select(-教师编号)
```

```
## # A tibble: 3 x 8
##   课程编号 课程名称 最高分 最低分 平均分 及格率 中等率 优秀率
##   <chr>    <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 01       语文         80     31   64.5  0.667  0.333  0
## 2 02       数学         90     30   72.7  0.833  0      0.167
## 3 03       英语         99     20   68.5  0.667  0      0.333
```

19. 按照各科成绩进行排序，并且显示排名

- 成绩表先根据课程编号分组，计算排名是用 rank()
- 排名时涉及到**对相同分数怎么排名**的问题，多个相同值在一起叫做” 结”，怎么处理” 结” 通常有六种方法
- 对成绩排名，一般是这样排：第 1 名，两个第 2 名，第 4 名 …… 这叫做用 “min” 方法处理 “结”
- 有两种实现，更简单的实现是直接用 min_rank()，来自” 小小不聪明” 的做法

```
score %>%
  group_by(课程编号) %>%
  mutate(排名 = min_rank(-成绩)) %>%   # 来自" 小小不聪明"
  arrange(课程编号, 排名)
```

```
## # A tibble: 18 x 4
## # Groups:   课程编号 [3]
##   学号  课程编号  成绩  排名
##   <chr> <chr>    <dbl> <int>
## 1 01    01          80     1
## 2 03    01          80     1
## 3 05    01          76     3
## 4 02    01          70     4
## 5 04    01          50     5
## 6 06    01          31     6
## # ... with 12 more rows
```

20. 查询学生的总成绩，并进行排名

- 类似上题 (略)

```r
score %>%
  group_by(学号) %>%
  summarise(总成绩 = sum(成绩)) %>%
  mutate(排名 = min_rank(-总成绩)) %>%
  arrange(排名)
```

```
## # A tibble: 7 x 3
##   学号  总成绩  排名
##   <chr>  <dbl> <int>
## 1 01       269     1
## 2 03       240     2
## 3 02       210     3
## 4 07       187     4
## 5 05       163     5
## 6 04       100     6
## # ... with 1 more row
```

21. 查询不同老师所教不同课程平均分从高到低显示

- 需要成绩、课程、教师信息，来自三个表，将它们连接起来
- 按教师、课程分组汇总计算平均分
- 成绩排名同上

```r
score %>%
  left_join(course, by = "课程编号") %>%
  left_join(teacher, by = "教师编号") %>%
  group_by(教师姓名, 课程名称) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  arrange(-平均成绩)
```

```
## # A tibble: 3 x 3
## # Groups:   教师姓名 [3]
##   教师姓名 课程名称 平均成绩
##   <chr>    <chr>       <dbl>
## 1 张三     数学         72.7
## 2 王五     英语         68.5
## 3 李四     语文         64.5
```

22. 查询所有课程的成绩第 2 至 3 名的学生信息及该课程成绩

- 分组修改：按课程分组，计算成绩排名，则为每门课程的成绩排名
- 筛选行：排名为 2 或 3

- 排序行：按课程、排名排序让结果更整齐
- 左连接：学生信息，需要连接学生表进来
- 选择列：删除不想显示的列

```
score %>%
  group_by(课程编号) %>%
  mutate(排名 = rank(-成绩, ties.method = "first")) %>%
  filter(排名 %in% 2:3) %>%
  arrange(课程编号, 排名) %>%
  left_join(student, by = " 学号") %>%
  select(-c(课程编号, 排名))
```

```
## # A tibble: 6 x 6
## # Groups:   课程编号 [3]
##   课程编号 学号   成绩 姓名  生日       性别
##   <chr>    <chr> <dbl> <chr> <chr>      <chr>
## 1 01       03       80 孙风  1990-05-20 男
## 2 01       05       76 周梅  1991-12-01 女
## 3 02       07       89 郑竹  1989-07-01 女
## 4 02       05       87 周梅  1991-12-01 女
## 5 03       07       98 郑竹  1989-07-01 女
## 6 03       02       80 钱电  1990-12-21 男
```

23. 统计各科成绩各分数段人数：课程编号，课程名称，[85,100]，[70,85)，[60,70)，[0,60) 及所占百分比

- 分组修改：按课程分组，用 cut 将成绩离散化为分数段
- 按课程、分数段分组计数
- 修改列：人数除以各科人数之和，计算百分比，注意当前分组变量是课程编号，sum(n) 就是对同一课程下的人数加和
- 左连接：需要课程信息，将课程表连接进来
- 选择列：删除不想显示的列

```
score %>%
  group_by(课程编号) %>%
  mutate(分数段 = cut(成绩, breaks = c(0,60,70,85,100), right = FALSE)) %>%
  count(课程编号, 分数段) %>%
  mutate(百分比 = scales::percent(n / sum(n), accuracy = 0.01)) %>%
  left_join(course, by = " 课程编号") %>%
  select(-教师编号)
```

```
## # A tibble: 9 x 5
```

```
## # Groups:   课程编号 [3]
##   课程编号 分数段       n 百分比 课程名称
##   <chr>    <fct>    <int> <chr>  <chr>
## 1 01       [0,60)       2 33.33% 语文
## 2 01       [70,85)      4 66.67% 语文
## 3 02       [0,60)       1 16.67% 数学
## 4 02       [60,70)      1 16.67% 数学
## 5 02       [70,85)      1 16.67% 数学
## 6 02       [85,100)     3 50.00% 数学
## # ... with 3 more rows
```

24. 查询学生的平均成绩及名次

- 分组汇总：按学号分组汇总每个学生的平均成绩
- 按平均成绩计算排名并排序，同前文

```
score %>%
  group_by(学号) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  mutate(排名 = min_rank(-平均成绩)) %>%
  arrange(排名)
```

```
## # A tibble: 7 x 3
##   学号  平均成绩  排名
##   <chr>    <dbl> <int>
## 1 07        93.5     1
## 2 01        89.7     2
## 3 05        81.5     3
## 4 03        80       4
## 5 02        70       5
## 6 04        33.3     6
## # ... with 1 more row
```

25. 查询各科成绩前三名的记录

- 分组筛选行：根据课程分组，做取最大行切片

```
score %>%
  group_by(课程编号) %>%
  slice_max(成绩, n = 3)
```

```
## # A tibble: 10 x 3
## # Groups:   课程编号 [3]
```

```
##   学号  课程编号  成绩
##   <chr> <chr>    <dbl>
## 1 01    01          80
## 2 03    01          80
## 3 05    01          76
## 4 01    02          90
## 5 07    02          89
## 6 05    02          87
## # ... with 4 more rows
```

26. 查询每门课被选修的学生数

- 按课程分组计数，即计算学生人数

```
score %>%
  count(课程编号)
```

```
## # A tibble: 3 x 2
##   课程编号     n
##   <chr>    <int>
## 1 01           6
## 2 02           6
## 3 03           6
```

27. 查询出只有两门课程的全部学生的学号和姓名

- 按学号分组计数，即每个学号所学的课程数，选出为 2 的行
- 半连接合并相应的学生信息进来，就是根据在右表，筛选左表的行
- 选择想要的列

```
score %>%
  count(学号) %>%
  filter(n == 2) %>%
  semi_join(student, ., by = " 学号") %>%
  select(学号, 姓名)
```

```
## # A tibble: 3 x 2
##   学号  姓名
##   <chr> <chr>
## 1 05    周梅
## 2 06    吴兰
## 3 07    郑竹
```

28. 查询男女生人数

- 按性别分组计数

```
student %>%
  count(性别)
```

```
## # A tibble: 2 x 2
##   性别      n
##   <chr> <int>
## 1 男        4
## 2 女        4
```

29. 查询名字中含有风字的学生信息

- 根据条件：姓名是否包含”风”，筛选行

```
student %>%
  filter(str_detect(姓名, "风"))
```

```
## # A tibble: 1 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 03    孙风  1990-05-20 男
```

30. 查询同姓名同性别的学生名单，并统计同姓名人数

- 按姓名、性别分组计数，数目大于 1 说明二者出现重复
- 同理，选出姓名重复，有几行就是有几个同姓名

```
student %>%
  count(姓名, 性别) %>%
  filter(n > 1)
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: 姓名 <chr>, 性别 <chr>, n <int>
```

```
student %>%
  count(姓名) %>%
  filter(n > 1) %>%
  summarise(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     0
```

31. 查询 1990 年出生的学生信息

- 根据条件：生日的年份等于 1990, 筛选行

```
student %>%
  filter(year(生日) == 1990)
```

```
## # A tibble: 5 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
## 5 08    王菊  1990-01-20 女
```

32. 计算每门课程的平均成绩，并按降序排列；若平均成绩相同，按课程编号升序排列

- 分组汇总 + 排序

```
score %>%
  group_by(课程编号) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  arrange(-平均成绩, 课程编号)
```

```
## # A tibble: 3 x 2
##   课程编号 平均成绩
##   <chr>       <dbl>
## 1 02           72.7
## 2 03           68.5
## 3 01           64.5
```

33. 查询平均成绩 85 分的学生学号、姓名和平均成绩

- 先按学号分组汇总计算平均成绩
- 根据平均成绩 85 分，筛选行
- 再左连接，将学生信息合并进来，选择想要的列

```
score %>%
  group_by(学号) %>%
  summarise(平均成绩 = mean(成绩)) %>%
  filter(平均成绩 >= 85) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 平均成绩)
```

```
## # A tibble: 2 x 3
##   学号  姓名  平均成绩
##   <chr> <chr>    <dbl>
## 1 01    赵雷      89.7
## 2 07    郑竹      93.5
```

34. 查询课程名称为数学，且分数低于 60 的学生姓名和分数

- 成绩表左连接，将课程信息合并进来
- 根据条件：课程名称等于数学且分数低于 60, 筛选行
- 左连接，将学生信息合并进来，选择想要的列

```
score %>%
  left_join(course, by = " 课程编号") %>%
  filter(课程名称 == " 数学", 成绩 < 60) %>%
  left_join(student, by = " 学号") %>%
  select(姓名, 成绩)
```

```
## # A tibble: 1 x 2
##   姓名  成绩
##   <chr> <dbl>
## 1 李云     30
```

35. 查询所有学生的课程及分数情况

- 成绩表左连接，将课程信息合并进来，删除没用的列
- 宽变长，变成每个学号占一行，每科成绩占一列
- 右连接学生信息，相当于以学生表为左表，选择想要的列
- 修改列：按行加和，忽略 NA, 计算每个学生的总分
  - pmap_dbl() 逐行迭代返回浮点向量
  - .[3:5] 是从数据框中选择列范围
  - ... 代表每行的三科成绩，需要用 c() 打包到一起
  - 忽略 NA, 用 sum 求和

```
score %>%
  left_join(course, by = " 课程编号") %>%
  select(-c(课程编号, 教师编号)) %>%
  pivot_wider(names_from = 课程名称, values_from = 成绩) %>%
  right_join(student, by = " 学号") %>%
  select(学号, 姓名, 语文, 数学, 英语) %>%
  mutate(总分 = pmap_dbl(.[3:5], ~ sum(c(...), na.rm = TRUE)))
```

```
## # A tibble: 8 x 6
```

```
##   学号  姓名   语文  数学  英语  总分
##   <chr> <chr> <dbl> <dbl> <dbl> <dbl>
## 1 01    赵雷     80    90    99   269
## 2 02    钱电     70    60    80   210
## 3 03    孙风     80    80    80   240
## 4 04    李云     50    30    20   100
## 5 05    周梅     76    87    NA   163
## 6 06    吴兰     31    NA    34    65
## # ... with 2 more rows
```

36. 查询任何一门课程成绩都在 70 分以上的姓名、课程名称和分数

- 前面步骤同 35 题
- 根据多列值构造筛选条件：所有成绩都大于 70 分，正常是用 `filter(if_all(2:4, ~ .x > 70))`，这里想忽略 NA 再做判断，采用的是逐行迭代的逻辑：
  - `pmap_lgl()` 逐行迭代返回逻辑向量
  - `.[2:4]` 是从数据框中选择列范围
  - `...` 代表每行的三科成绩，需要用 `c()` 打包到一起
  - 忽略 NA 判断 `>70`，再取 `all()`
- 左连接，将学生信息合并进来，再选择想要的列

```
score %>%
  left_join(course, by = " 课程编号") %>%
  select(-c(课程编号, 教师编号)) %>%
  pivot_wider(names_from = 课程名称, values_from = 成绩) %>%
  filter(pmap_lgl(.[2:4], ~ all(na.omit(c(...)) > 70))) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 语文, 数学, 英语)
```

```
## # A tibble: 4 x 5
##   学号  姓名   语文  数学  英语
##   <chr> <chr> <dbl> <dbl> <dbl>
## 1 01    赵雷     80    90    99
## 2 03    孙风     80    80    80
## 3 05    周梅     76    87    NA
## 4 07    郑竹     NA    89    98
```

37. 查询不及格的课程

- 根据不及格条件筛选行
- 左连接，将课程信息合并进来
- 选择想要的列

```r
score %>%
  filter(成绩 < 60) %>%
  left_join(course, by = " 课程编号") %>%
  select(课程编号, 课程名称, 成绩)
```

```
## # A tibble: 5 x 3
##   课程编号 课程名称   成绩
##   <chr>    <chr>     <dbl>
## 1 01       语文         50
## 2 02       数学         30
## 3 03       英语         20
## 4 01       语文         31
## 5 03       英语         34
```

38. 查询课程 01 的成绩大于等于 80 的学生学号和姓名

- 根据课程条件筛选行
- 左连接，将学生信息合并进来
- 选择想要的列

```r
score %>%
  filter(课程编号 == "01", 成绩 >= 80) %>%
  left_join(student, by = " 学号") %>%
  select(学号, 姓名, 成绩)
```

```
## # A tibble: 2 x 3
##   学号  姓名    成绩
##   <chr> <chr> <dbl>
## 1 01    赵雷     80
## 2 03    孙风     80
```

39. 同 26. (略)

40. 查询选修” 张三” 老师所授课程的学生中，成绩最高的学生信息及其成绩

- 教师表左连接，将课程信息合并进来
- 根据教师姓名等于” 张三”，筛选行，得到张三所授课程，将作为右表
- 与成绩表做半连接，即根据右表筛选左表，得到张三所授课程的成绩
- 对成绩，取最大做行切片，找到成绩最高的学号
- 左连接，将这些学号的学生信息合并进来

```r
teacher %>%
  left_join(course, by = " 教师编号") %>%
```

```
  filter(教师姓名 == " 张三") %>%
  semi_join(score, ., by = " 课程编号") %>%
  slice_max(成绩) %>%
  left_join(student, by = " 学号")
```

```
## # A tibble: 1 x 6
##   学号  课程编号  成绩 姓名  生日       性别
##   <chr> <chr>    <dbl> <chr> <chr>      <chr>
## 1 01    02          90 赵雷  1990-01-01 男
```

41. 查询不同课程成绩相同的学生学号、课程编号、学生成绩

- 成绩表按课程分组，对成绩计数，统计的是每门课程各个成绩出现的次数
- 筛选次数 >1 的行，就选出了成绩相同的课程，将作为右表
- 再与成绩表做半连接，实现根据右表筛选左表
- 为了结果更直观，按课程编号排序

```
score %>%
  group_by(课程编号) %>%
  count(成绩) %>%
  filter(n > 1) %>%
  semi_join(score, ., by = c(" 课程编号", " 成绩")) %>%
  arrange(课程编号)
```

```
## # A tibble: 4 x 3
##   学号  课程编号  成绩
##   <chr> <chr>    <dbl>
## 1 01    01          80
## 2 03    01          80
## 3 02    03          80
## 4 03    03          80
```

42. 查询每门课程成绩最好的前两名

- 类似 25 题

```
score %>%
  group_by(课程编号) %>%
  slice_max(成绩, n = 2)
```

```
## # A tibble: 6 x 3
## # Groups:   课程编号 [3]
##   学号  课程编号  成绩
```

```
##   <chr> <chr>    <dbl>
## 1 01    01          80
## 2 03    01          80
## 3 01    02          90
## 4 07    02          89
## 5 01    03          99
## 6 07    03          98
```

43. 统计每门课程的学生选修人数（超过 5 人的课程才统计）。要求输出课程号和选修人数，查询结果按人数降序排列，若人数相同，按课程号升序排列

- 按课程分组计数，得到每门课程的选修人数，接着筛选出大于 5 的行
- 按人数降序、课程号升序做行排序

```
score %>%
  count(课程编号) %>%
  filter(n > 5) %>%
  arrange(-n, 课程编号)
```

```
## # A tibble: 3 x 2
##   课程编号     n
##   <chr>    <int>
## 1 01           6
## 2 02           6
## 3 03           6
```

44. 检索至少选修两门课程的学生学号

- 按学号分组计数，就是每个学号的课程数
- 根据条件筛选行

```
score %>%
  count(学号) %>%
  filter(n >= 2)
```

```
## # A tibble: 7 x 2
##   学号      n
##   <chr> <int>
## 1 01        3
## 2 02        3
## 3 03        3
## 4 04        3
## 5 05        2
```

```
## 6 06        2
## # ... with 1 more row
```

45. 查询选修了全部课程的学生信息

```
course %>%
  left_join(score, by = " 课程编号") %>%
  select(-c(课程编号, 教师编号)) %>%
  pivot_wider(names_from = 课程名称, values_from = 成绩) %>%
  right_join(student, by = " 学号") %>%
  filter(if_all(2:4, ~ !is.na(.x))) %>%
  select(-(2:4))
```

```
## # A tibble: 4 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <chr>      <chr>
## 1 01    赵雷  1990-01-01 男
## 2 02    钱电  1990-12-21 男
## 3 03    孙风  1990-05-20 男
## 4 04    李云  1990-08-06 男
```

46. 查询各学生的年龄：按照出生日期来算，当前月日 < 出生年月的月日，则年龄减 1

- 修改列：将生日转化为日期型
- 年龄就是生日至今天的时间段，整除单位 1 年，等于有几个 1 年

```
student %>%
  mutate(生日 = ymd(生日),
         年龄 = interval(生日, today()) %/% dyears(1))
```

```
## # A tibble: 8 x 5
##   学号  姓名  生日       性别   年龄
##   <chr> <chr> <date>     <chr> <dbl>
## 1 01    赵雷  1990-01-01 男       32
## 2 02    钱电  1990-12-21 男       31
## 3 03    孙风  1990-05-20 男       31
## 4 04    李云  1990-08-06 男       31
## 5 05    周梅  1991-12-01 女       30
## 6 06    吴兰  1992-03-01 女       29
## # ... with 2 more rows
```

47. 查询本周过生日的学生

- 修改列：将生日转化为日期型
- 根据条件：生日周等于本周，筛选行

```
student %>%
  mutate(生日 = ymd(生日)) %>%
  filter(week(生日) == week(today()))
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: 学号 <chr>, 姓名 <chr>, 生日 <date>, 性别 <chr>
```

48. 查询下周过生日的学生

- 类似 47 题

```
student %>%
  mutate(生日 = ymd(生日)) %>%
  filter(week(生日) == week(today()) + 1)
```

```
## # A tibble: 1 x 4
##   学号  姓名  生日       性别
##   <chr> <chr> <date>     <chr>
## 1 06    吴兰  1992-03-01 女
```

49. 查询本月过生日的学生

- 类似 47 题

```
student %>%
  mutate(生日 = ymd(生日)) %>%
  filter(month(生日) == month(today()))
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: 学号 <chr>, 姓名 <chr>, 生日 <date>, 性别 <chr>
```

50. 查询下月过生日的学生

- 类似 47 题

```
student %>%
  mutate(生日 = ymd(生日)) %>%
  filter(month(生日) == month(today()) + 1)
```

```
## # A tibble: 1 x 4
##   学号  姓名  生日       性别
```

```
##   <chr> <chr> <date>     <chr>
## 1 06    吴兰  1992-03-01 女
```