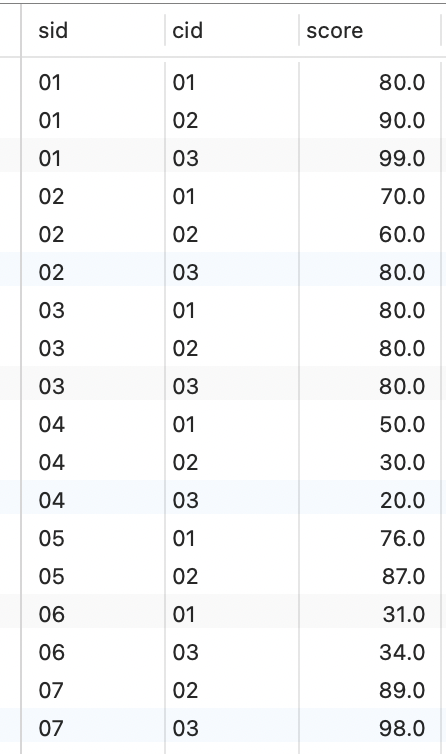
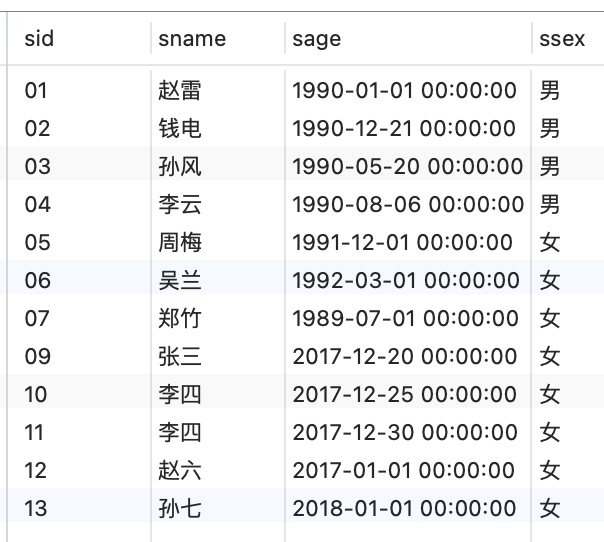
案例展示

# 一 SQL查询

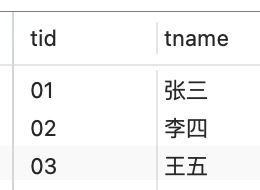
数据表1：成绩表，表名：sc



数据表2：学生表，表名：student



数据表3：教师表，表名：teacher



数据表4：课程表，表名：course

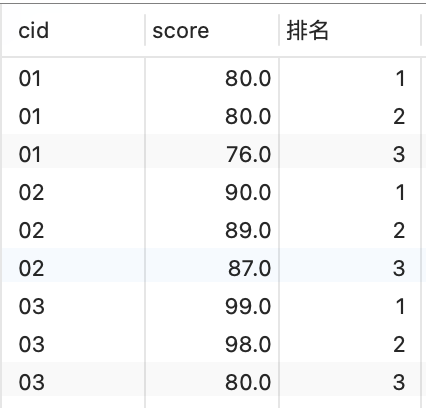


## 查询各科成绩前三名的记录

1.1:方法一

|  |
| --- |
| select  cid,  score,  rn '排名'  from (  select  sid,  cid,  score,  row\_number() over(partition by cid order by score desc) rn  from sc  ) a  where 1=1  and rn <3 |

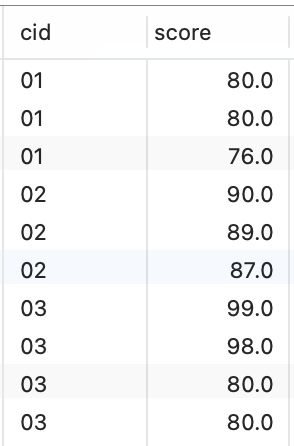
运行结果：



1.2:方法二

|  |
| --- |
| select  cid,  score  from  sc  where 1=1  and (  select  count(\*)  from  sc as a  where 1=1  and sc.cid=a.cid and sc.score<a.score)<3  order by  sc.cid asc,  sc.score desc |

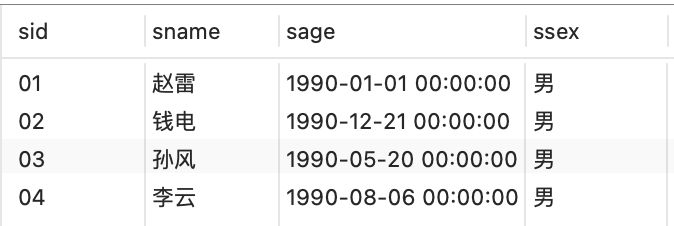
运行结果：



## 查询和" 01 "号同学学习课程完全相同的其他同学信息

|  |
| --- |
| select  student.\*  from  student  where 1=1  and student.sid not in (  select  t2.sid  from (  select  student.sid,  t.cid  from  student,  select cid from sc where sid='01')t  ) t2  left join sc  on t2.sid=sc.sid  and t2.cid=sc.cid  where sc.sid is null) |

运行结果：

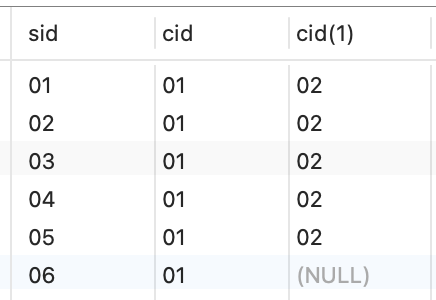


## 查询存在" 01 "课程但可能不存在" 02 "课程的情况

(不存在时显示为 null )

|  |
| --- |
| select  t.sid,  t.cid,  t1.cid  from (  select  \*  from  sc  where 1=1  and cid='01')t  left join (  select  \*  from  sc  where 1=1  and cid='02')t1  on t.sid=t1.sid |

运行结果：

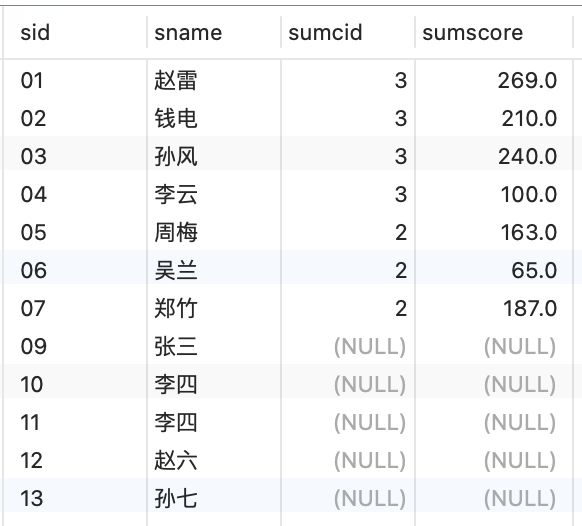


## 查询所有同学的编号、姓名、选课数、总成绩

(没成绩的显示为 null )

|  |
| --- |
| select  student.sid,  student.sname,  t.sumcid,  t.sumscore  from student  left join(  select  sid,  count(cid) sumcid,  sum(score) sumscore  from sc  where 1=1  group by sid) t  on student.sid=t.sid |

运行结果：

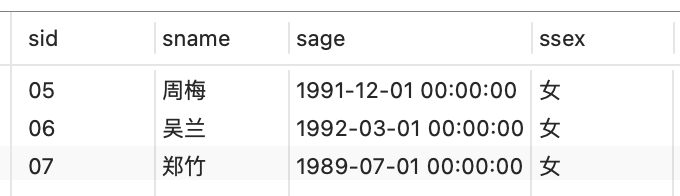


## 查询没学全所有课程的同学的信息,

不包括从未选课的学生

|  |
| --- |
| select  student.\*  from  student,(  select  sc.sid,  count(sc.cid) countcid  from sc  group by sc.sid  having countcid<(select count(\*) from course)) t  where student.sid=t.sid |

运行结果：



## 6、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 ID，最高分，最低分，平均分，选修人数，及格率，中等率，优良率，优秀率

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

-- 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

|  |
| --- |
| select  sc.cid,  max(sc.score)as 最高分,  min(sc.score)as 最低分,  avg(sc.score)as 平均分,  count(\*)as 选修人数,  sum(case when sc.score>=60 then 1 else 0 end)/count(\*)as 及格率,  sum(case when sc.score>=70 and sc.score<80 then 1 else 0 end)/count(\*)as 中等率,  sum(case when sc.score>=80 and sc.score<90 then 1 else 0 end)/count(\*)as 优良率,  sum(case when sc.score>=90 then 1 else 0 end)/count(\*)as 优秀率  from sc  where 1=1  group by sc.cid  order by count(\*) desc,sc.cid asc |

运行结果：



## 7、统计各科成绩各分段占比

课程编号，课程名称，[100-85]，[85-70]，[70-60]，[60-0] 及所占百分比

|  |
| --- |
| select  t.cid,  course.cname,  t.优秀,  t.良好,  t.合格,  t.普通  from course,(  select sc.cid,  concat(sum(case when sc.score<100 and sc.score>=85 then 1 else 0 end)/count(\*)\*100,'%')as '优秀',  concat(sum(case when sc.score<85 and sc.score>=70 then 1 else 0 end)/count(\*)\*100,'%')as '良好',  concat(sum(case when sc.score<70 and sc.score>=60 then 1 else 0 end)/count(\*)\*100,'%')as '合格',  concat(sum(case when sc.score<60 then 1 else 0 end)/count(\*)\*100,'%')as '普通'  from sc  group by sc.cid)as t  where 1=1  and course.cid=t.cid |

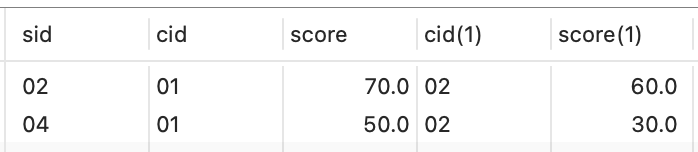
运行结果：



## 8、查询" 01 "课程比" 02 "课程成绩高的学生信息及课程分数

|  |
| --- |
| select  t.\*,  t1.cid,  t1.score  from  (select \* from sc where cid='01')t,  (select \* from sc where cid='02')t1  where 1=1  and t.sid=t1.sid  and t.score>t1.score |

运行结果：



## 9、查询没学过"张三"老师讲授的任一门课程的学生姓名

|  |
| --- |
| select  student.\*  from  student  where student.sid not in (  select  student.sid  from  student,  sc,  teacher,  course  where 1=1  and student.sid=sc.sid  and sc.cid=course.cid  and course.tid=teacher.tid  and teacher.tname = '张三') |

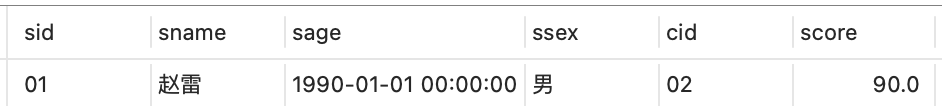
运行结果：

## 10、查询选修「张三」老师课程的成绩最高学生及成绩

要求成绩不重复

|  |
| --- |
| select  student.\*,  sc.cid,  sc.score  from  sc,  student,  teacher,  course  where 1=1  and sc.sid=student.sid  and sc.cid=course.cid  and teacher.tid=course.tid  and teacher.tname='张三'  order by sc.score desc  limit 1 |

运行结果：



# 一 数据分析案例

## 房价可视化分析

对房价最高的10城的房价做分析，用可视化展示平均价/同比/环比进行展示

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  plt.rcParams['figure.figsize']=(8.0,5.0)#固定显示大小  plt.rcParams['font.family']=['sans-serif']#设置中文字体  plt.rcParams['font.sans-serif']=['SimHei']#设置中文字体  plt.rcParams['axes.unicode\_minus']=False#显示负号  dfr=pd.read\_excel('cityhouseprice.xlsx')#读取原数据  df=(#对数据做清洗  dfr.assign(同比=dfr.同比.str[:-1])#.astype(float)#去百分号并转为浮点型  .assign(环比=dfr.环比.str[:-1])#.astype(float))#去百分号并转为浮点型  .loc[:,['城市名称','平均单价','同比','环比']]#重命名列  )  df=df.astype({'同比':float}).astype({'环比':float})#类型转换  df=df.set\_index('城市名称').平均单价#设置索引并取出索引和平均单价列  df=df.plot.bar()#生成柱状图  运行结果如下图：（汉字字符显示有问题待解决）    (#对同环比生成表示正负值的条形图  df.set\_index('城市名称')  .loc[:,'同比':'环比']  .plot.bar()  )  运行结果如下图：（汉字字符显示有问题待解决）    (#用颜色样式标注的方式显示同环比值变化较大的内容，极值用红色标识  df.style.highlight\_max(color='red',subset=['同比','环比'])  .highlight\_min(subset=['同比','环比'])  .format({'平均单价':"{:,.0f}"})  .format({'同比':"{:2}%",'环比':"{:2}%"})  )  运行结果如下图：    (#用倒序方式对指定列显示不同宽度的背景色  df.style  .bar(subset=['平均单价'],color='yellow')#指定列及颜色  )  运行结果如下图：    (#在表格中对单价用颜色标识，对同环比用指定颜色根据值的大小填充背景  df.style.background\_gradient(subset=['平均单价'],cmap='BuGn')#指定色系  .format({'同比':"{:2}%",'环比':"{:2}%"})#指定显示的小数位  .bar(subset=['同比'],#设置同比的显示样式  color=['#ffe4e4','#bbf9ce'],#指定上涨、下降的颜色  vmin=0,vmax=15,#范围定为以0为基准的上下15  align='zero'#设置0值居中  )  .bar(subset=['环比'],#设置环比的显示样式  color=['red','green'],#指定上涨、下降的颜色  vmin=0,vmax=11,#范围定为以0为基准的上下20  align='zero'#设置0值居中  )  )  运行结果如下图： |

## 对2020年疫情发展期分析

了解一下它的累积确证、新增高峰、世界死亡前10国家、中美病例数趋势对比、中美死亡率对比等变化情况。

|  |
| --- |
| #构建数据  import pandas as pd  df=pd.read\_csv('countries-aggregated.csv')#读取原文件  df.tail(5)  构建数据如下：    #查看china确诊人数累积增长曲线  (#查看china确诊人数累积增长曲线  df.loc[df.Country=='China']#只选china的  .set\_index('Date')#日期为索引  .Confirmed#看确诊的  .plot()#画折线图  )  运行结果如下图：    #查询china的确诊人数每日新增数量  (  df.loc[df.Country=='China']  .set\_index('Date')#设置日期为索引  .assign(new=lambda x:x.Confirmed.diff())#增加一个每日新增数量列  .new#只取新增的数量列  .plot()#绘折线图  )  运行结果如下图：    #找出确诊病例在一万以上且死亡率前十的  (  df.loc[df.Date==df.Date.max()]#由于是累积数据，所以需要看取最新的日期  .loc[df.Confirmed>10000]#确认10000人以上  .assign(rate=lambda x:x.Deaths/x.Confirmed)#增加死亡率的列  .sort\_values('rate',ascending=False)#按死亡率最高排序  .set\_index('Country')#以国家名称为索引  .head(10)#取前10个  .rate#选取死亡率  .sort\_values(ascending=True)#为了图形的直观性，按降序排列  .plot.barh()#barh为获取到横向柱状图  )  运行结果如下图：    #查看China和Us两国的确诊数量增长趋势  (  df.loc[df.Country.isin(['China','US']),['Country','Date','Confirmed']]#只取想要的国家和数据  .groupby(['Country','Date'])#分组  .max()#取分组后的确诊人数最大值数据行  .unstack()#展开让纵向变横向  .T#转置横纵向维度，让维度变成日期和国家的确诊人数  .droplevel(0)#删除转置后的无用索引  .plot()#绘折线图  )  运行结果如下图：    #查看China和Us两国的死亡率对比  (  df.loc[(df.Country.isin(['China','US']))&(df.Date==df.Date.max())]#只要这两个国家的最新数据  .assign(rate=df.Deaths/df.Confirmed)#增加死亡率的数据列和数据  .set\_index('Country')#将国家设置为索引  .rate#只取国家索引后的死亡率的列  .plot.bar()#绘制成条形图  )  运行结果如下图：    #两个国家的死亡率趋势对比  (  df.loc[(df.Country.isin(['China','US']))]#只要这两个国家的最新数据  .assign(rate=df.Deaths/df.Confirmed)#增加死亡率数据列和数据  .groupby(['Country','Date'])#按国家和日期分组  .max()#取得分组后的最大值数据行  .rate#只取两个国家在各个日期的死亡率数据  .unstack()#展开为按日期和国家做横纵向  .T#对横纵向数据进行转换  .plot()#绘制出死亡率对比折线图  )  运行结果如下图： |

## 推算三天中两天下雨的概率

天气预报说，在今后的三天中，每一天下雨的概率均为40%，请问这三天中恰有两天下雨的概率是多少？

|  |
| --- |
| import pandas as pd  import numpy as np  rangenp=np.random.default\_rng()#定义随机对象  days=100000#随机天数  arr=rangenp.integers(0,1000,days)#生成随机数字  (  pd.DataFrame()  .assign(x=arr)#将随机数字增加到列  .astype(str)#转为字符  .assign(x=lambda d:d.x.str.zfill(3))#在不足三位数字前补0  .assign(a=lambda d:d.x.str.count(r'1|2|3|4'))#统计0～9共10个数字串中1～4（40%下雨）的数量  .query('a==2')#筛选出两天下雨的数据  )  len(\_)/days#用两天会下雨的行数除以总行数得出三天中有两天下午的概率  运行结果为0.28757（**即28%**）如下图： |

## RFM用户分层

评估用户消费能力、衡量用户贡献价值的重要工具。RFM代表的是最近一次消费时间间隔（Recency）、消费频率（Frequency）和消费金额（Monetary），便于精细化运营。

|  |
| --- |
| import pandas as pd  import faker  #构造数据  f=faker.Faker('zh-cn')  df=pd.DataFrame({'用户':[f.name()for i in range(20000)],#生成两万个消费用户  '购买日期':[f.date\_between(start\_date='-1y',  end\_date='today')for i in range(20000)],#生成两万个截止今天为止的一年内日期作为消费日期  '金额':[f.random\_int(10,100)for i in range(20000)]})#生成两万个10到100的数字作为消费金额  df=df.astype({'购买日期':'datetime64[ns]'})#类型转换  #构造数据结果如下：    #构建用户的九大分类  label\_names={  111:'重要价值用户',  110:'一般价值用户',  101:'重要发展用户',  100:'一般发展用户',  11:'重要保持用户',  10:'一般保持用户',  1:'重要挽留用户',  0:'一般挽留用户'}  (#集散处分层数据并用统计图展现  pd.DataFrame({'r':r,'f':f,})#构建r和f列  .assign(m=lambda x:df.groupby(['用户']).sum()['金额']/x.f)#计算出m列值并追加该列  .assign(r\_e=lambda x:(x.r>x.r.mean())\*1)#做归一化操作，当一行中的r大于当列平均值即赋值1  .assign(f\_e=lambda x:(x.f>x.f.mean())\*1)#做归一化操作，当一行中的f大于当列平均值即赋值1  .assign(m\_e=lambda x:(x.m>x.m.mean())\*1)#做归一化操作，当一行中的m大于当列平均值即赋值1  .assign(label=lambda x:x.r\_e\*100+x.f\_e\*10+x.m\_e\*1)#将三个新列连接起来  .assign(label\_names=lambda x:x.label.map(label\_names))#用map对用户做归类  .groupby('label').count().r.plot.bar()#对每一类用户进行统计并生成柱状图  )  运行结果如下图：通过结果得知重要发展用户及一般发展用户居多 |

## 计算gmv变化

一份两个年度同期的营收数据，有两个数据列，一个为日期，另一个为对应日期的交易金额（GMV）。业务方的需求是想知道2020年相对2019年同期的金额是上涨还是下跌。

|  |
| --- |
| import pandas as pd  import numpy as np  #数据准备  list1 = np.random.randint(50,80,366).tolist()#生成一个包含366各随机数且范围在50-80的列表  d1f = pd.DataFrame(list1)#将列表赋值给一个pandas类型DataFrame的变量d1f  list2 = np.random.randint(70,100,365).tolist()#生成一个包含365各随机数且范围在70-100的列表  d2f = pd.DataFrame(list2)#将列表赋值给一个pandas类型DataFrame的变量d2f  t1=pd.Timestamp('2020-01-11')#给定一个日期  t1.replace(day=1)#将日期替换为当月第一天  index1=pd.date\_range(start=t1,end=(t1+pd.offsets.YearEnd()))#生成2020年的全部日期数据  df1=pd.DataFrame(index1,columns=["date"])#生成DataFrame数据并指定列名  df1.loc[:,'gmv']=d1f#追加并指定列名  df1=df1.astype({'date':'datetime64[ns]'})#类型转换  t2=pd.Timestamp('2021-01-11')#给定一个日期  t2.replace(day=1)#将日期替换为当月第一天  index2=pd.date\_range(start=t2,end=(t2+pd.offsets.YearEnd()))#生成2021年的全部日期数据  df2=pd.DataFrame(index2,columns=["date"])#生成DataFrame数据并指定列名  df2.loc[:,'gmv']=d2f#追加列且指定列名  df2=df2.astype({'date':'datetime64[ns]'})#类型转换  df=pd.concat([df1,df2])#连接两个数据列  #对数据清洗和获取备用  df=df.reset\_index()#重置为自然索引  df=df[['date','gmv']]#只取指定的两列  df=df.set\_index('date')#设置索引  df=df.groupby([lambda x:x.month,lambda x:x.day]).apply(lambda x:x.gmv)#按月和日分组并去除分组数据  df=df.diff()#计算下行减上行的值  df=df.dropna()#删除有空值的行  #去除数据并生成图表  df=df.loc[lambda x:x.index.year==2021]#只取2021年的数据行  df=df.plot()#生成折线图  df |