数据准备

已知有如下4张表：

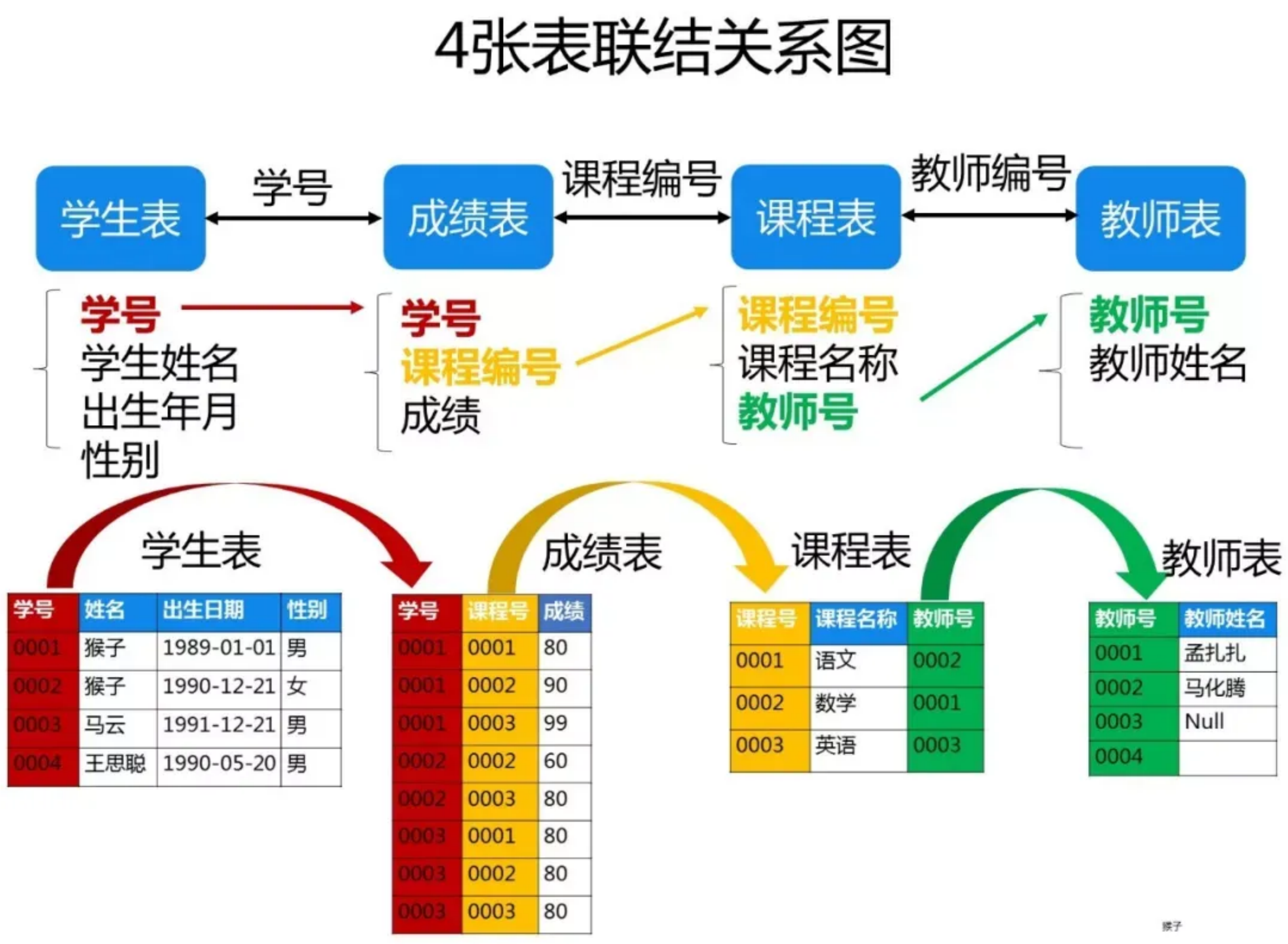
学生表：student(学号,学生姓名,出生日期,性别)

成绩表：score(学号,课程号,成绩)

课程表：course(课程号,课程名称,教师号)

教师表：teacher(教师号,教师姓名)

根据以上信息按照下面要求写出对应的SQL语句。



学号	姓名	出生日期	性别
0001	猴子	1989-01-01	男
0002	猴子	1990-12-21	女
0003	马云	1991-12-21	男
0004	王思聪	1990-05-20	男

学号	课程号	成绩
0001	0001	80
0001	0002	90
0001	0003	99
0002	0002	60
0002	0003	80
0003	0001	80
0003	0002	80
0003	0003	80

课程号	课程名称	教师号
0001	语文	0002
0002	数学	0001
0003	英语	0003

教师号	教师姓名
0001	孟扎扎
0002	马化腾
0003	Null
0004	

1、创建数据库和表

学生表

```
CREATE TABLE `student` (
  `id` int(4) ZEROFILL PRIMARY KEY COMMENT '学生ID',
  `name` varchar(255) COMMENT '学生名',
`birth` date COMMENT '出生日期',
`sex` CHAR ( 1 ) COMMENT '性别');
```

教师表

```
CREATE TABLE `teacher` (
  `id` int(4) ZEROFILL PRIMARY KEY COMMENT '教师号',
  `name` varchar(255) COMMENT '教师姓名');
```

课程表

```
CREATE TABLE `course` (
  `id` int(4) ZEROFILL PRIMARY KEY COMMENT '课程号',
  `name` varchar(255) COMMENT '课程名',
  `teacher_id` int(4) ZEROFILL COMMENT '教师号',
CONSTRAINT FOREIGN KEY ( `teacher_id` ) REFERENCES `teacher` ( `id` ));
```

成绩表

```
CREATE TABLE `score` (
  `student_id` int(4) ZEROFILL COMMENT '学号',
  `course_id` int(4) ZEROFILId COMMENT '课程号',
  `score` TINYINT UNSIGNED COMMENT '分数',
PRIMARY KEY ( `student_id`, `course_id` ),
CONSTRAINT FOREIGN KEY ( `student_id` ) REFERENCES `student` ( `id` ),
CONSTRAINT FOREIGN KEY ( `course_id` ) REFERENCES `course`(`id`));
```

2.向表中添加数据

学生表

```
# insert into student(学号,姓名,出生日期,性别)

insert into student(id,name,birth,sex)
values('0001' , '猴子' , '1989-01-01' , '男');

insert into student(id,name,birth,sex)
values('0002' , '猴子' , '1990-12-21' , '女');

insert into student(id,name,birth,sex)
values('0003' , '马云' , '1991-12-21' , '男');

insert into student(id,name,birth,sex)
values('0004' , '王思聪' , '1990-05-20' , '男');
```

教师表

```
-- 教师表：添加数据
```

```
insert into teacher(id,name)
values('0001' , '孟扎扎');

insert into teacher(id,name)
values('0002' , '马化腾');

-- 这里的教师姓名是空值（null）
insert into teacher(id,name)
values('0003' , null);

-- 这里的教师姓名是空字符串（''）
insert into teacher(id,name)
values('0004' , '');
```

课程表

```
insert into course(id,name,teacher_id)
values('0001' , '语文' , '0002');

insert into course(id,name,teacher_id)
values('0002' , '数学' , '0001');

insert into course(id,name,teacher_id)
values('0003' , '英语' , '0003');
```

成绩表

```
insert into score(student_id,course_id,score)
values('0001' , '0001' , 80);

insert into score(student_id,course_id,score)
values('0001' , '0002' , 90);

insert into score(student_id,course_id,score)
values('0001' , '0003' , 99);

insert into score(student_id,course_id,score)
values('0002' , '0002' , 60);

insert into score(student_id,course_id,score)
values('0002' , '0003' , 80);

insert into score(student_id,course_id,score)
values('0003' , '0001' , 80);

insert into score(student_id,course_id,score)
values('0003' , '0002' , 80);
```

```
insert into score(student_id,course_id,score)
values('0003' , '0003' , 80);
```

sql正题

1. 查询姓“猴”的学生名单

```
select * from student where `姓名` like '猴%'
```

2. 查询姓名中最后一个字是“猴”的学生名单

```
select * from student where `姓名` like '%猴'
```

3. 查询姓名中带“猴”的学生名单

```
select * from student where `姓名` like '%猴%'
```

4.查询姓“孟”老师的个数

```
select count(1) from teacher where `教师姓名` like '孟%'
```

5.查询课程编号为“0002”的总成绩

```
SELECT sum(`成绩`) FROM `score` where `课程号`='0002'
```

6. 查询已经选课的学生人数（可能存在一个同学选了多门课）

```
select count(DISTINCT 学号) from score;
```

7. 查询各科成绩最高和最低分，按课程号分组

```
select `课程号`, max(`成绩`),min(`成绩`) from score GROUP BY `课程号`
```

8. 查询每门课被选修的学生人数(别名 student_num)

```
select `课程号`, count(*) as student_num from score GROUP BY `课程号`
```

9. 查询男生、女生人数（别名 sex_num），并按性别分组

```
select `性别`, count(*) as sex_num  from student GROUP BY `性别`
```

10. 查询平均成绩大于60分学生的学号和平均成绩

```
select avg(`成绩`),`学号` from score GROUP BY `学号` HAVING avg(`成绩`)>60
```

11. 查询至少选修两门课程的学生学号

```
select count(`课程号`),`学号` from score GROUP BY `学号` HAVING count(`课程号`)>=2
```

12. 查询同名同姓学生名单并统计同名人数

```
select COUNt(`姓名`),`姓名` from student GROUP BY  `姓名` HAVING COUNt(`姓名`) > 1
```

13. 查询不及格的课程并按课程号从大到小排列

```
select * from score where `成绩` < 60 ORDER BY 课程号 desc
```

14. 查询每门课程的平均成绩（别名用 avg_score），结果按平均成绩升序排序，平均成绩相同时，按课程号降序排列

```
SELECT
  avg(成绩) AS 平均,课程号
FROM
  score
GROUP BY
  `课程号`
ORDER BY
  平均,
  课程号 DESC
```

15. 检索课程编号为“0004”且分数小于60的学生学号，结果按按分数降序排列

```
SELECT
  学号 ,成绩
FROM
  score
WHERE
  课程号 = '0004'
  AND 成绩 < 60
ORDER BY
  成绩 DESC
```

16. 统计每门课程的学生选修人数(超过2人的课程才统计),要求输出课程号和选修人数(别名用num)，查询结果按人数降序排序，若人数相同，按课程号升序排序

```
SELECT
  count(学号) AS 人数,
  `课程号`
FROM
  score
GROUP BY
  `课程号`
HAVING
  人数 > 2
ORDER BY
  人数 DESC,
  `课程号`
```

17. 查询两门以上不及格课程的同学的学号及其平均成绩(若95分以下为不及格)

```
SELECT
  学号,
  avg(成绩)
FROM
  score
WHERE
  学号 IN ( SELECT 学号 FROM score WHERE 成绩 < 95 GROUP BY 学号 HAVING count(*) > 1 )
GROUP BY
  学号

  /*
  SELECT
  student_id,
  avg(score)
FROM
  score
WHERE
  student_id IN ( SELECT student_id FROM score WHERE score < 95 GROUP BY student_id
HAVING count(*) > 1 )
GROUP BY
  student_id
  */
```

18. 查询学生的总成绩(别名用 sum_score)并进行排名

```
SELECT
  学号,
  sum(成绩) 总成绩
FROM
  score
GROUP BY
  学号
ORDER BY
  总成绩

  /*
  SELECT
  student_id,
  sum(score) AS sum_score
FROM
  score
GROUP BY
  student_id
ORDER BY
  sum_score
  */
```

19. 查询平均成绩大于60分的学生的学号和平均成绩（别名用avg_score）， 通过学号分组

```
SELECT
  学号,
  avg(成绩) 平均成绩
FROM
  score
GROUP BY
  学号
HAVING
  平均成绩 > 60
```

20. 查询所有课程成绩小于90分学生的学号、姓名

```
SELECT
  *
FROM
  student
WHERE
  学号 IN (
  SELECT
    学号
  FROM
    score
  GROUP BY
    学号
  HAVING
  max(成绩) < 90
  )

  /*
  SELECT
  *
FROM
  student
WHERE
  id IN (
  SELECT
    student_id
  FROM
    score
  GROUP BY
    student_id
  HAVING
  max(score) < 90
  )
  */
```

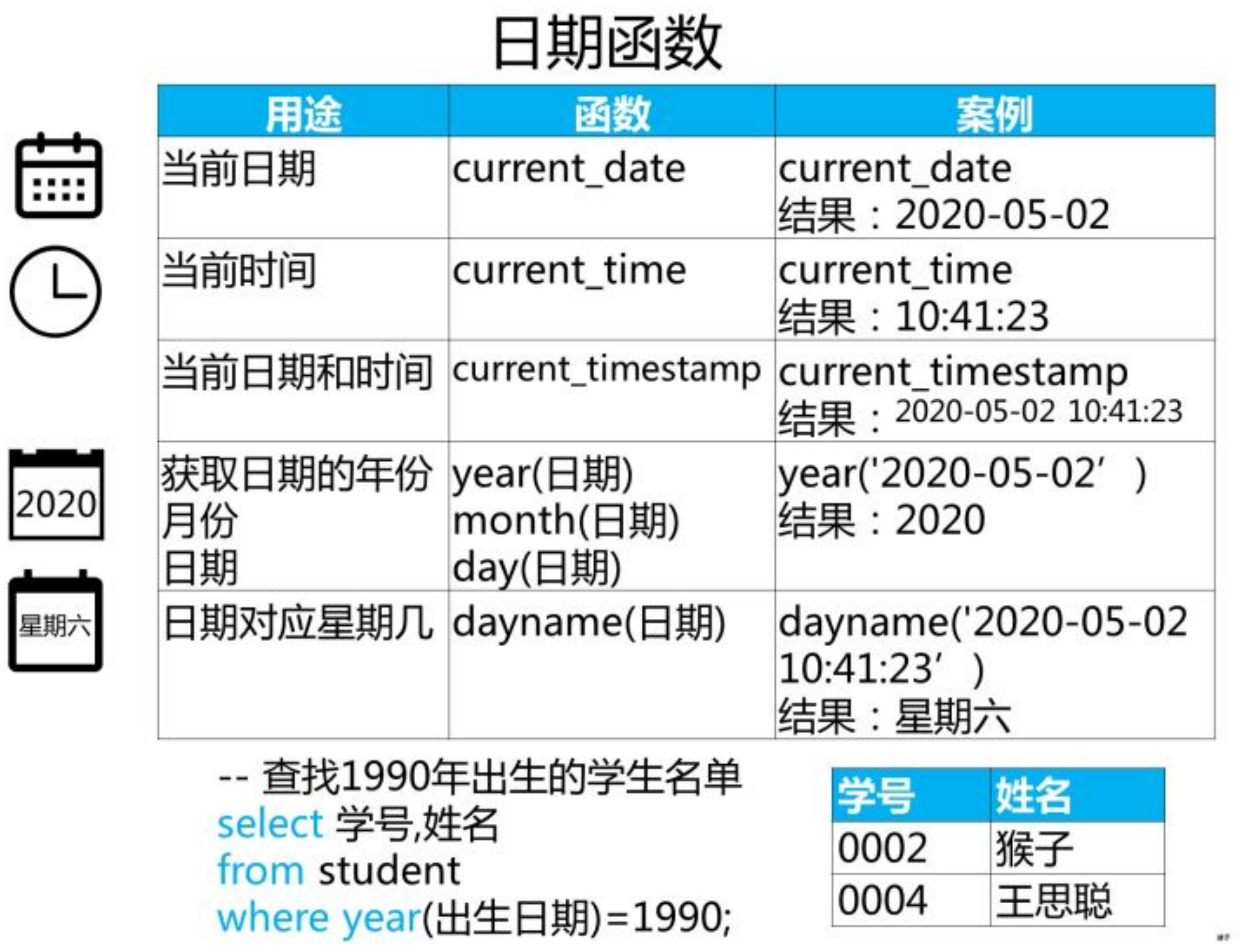
21. 查询没有选修所有课的学生的学号、姓名

```
SELECT
  学号,姓名
FROM
  student
WHERE
  学号 NOT IN (
  SELECT
    学号
  FROM
    score
  GROUP BY
    学号
  HAVING
  count(*) > 2
  )

  /*
  SELECT
  id,name
FROM
  student
WHERE
```

```
  id NOT IN (
  SELECT
    student_id
  FROM
    score
  GROUP BY
    student_id
  HAVING
  count(*) > 2
  )
  */
```

22. 查询出只选修了两门课程的全部学生的学号和姓名

```
SELECT
  学号,姓名
FROM
  student
WHERE
  学号  IN (
  SELECT
    学号
  FROM
    score
  GROUP BY
    学号
  HAVING
  count(*) = 2
  )

  /*
  SELECT
  id,name
FROM
  student
WHERE
  id  IN (
  SELECT
    student_id
  FROM
    score
  GROUP BY
    student_id
  HAVING
  count(*) = 2
  )
  */
```



日期函数

用途	函数	案例
当前日期	current_date	current_date 结果：2020-05-02
当前时间	current_time	current_time 结果：10:41:23
当前日期和时间	current_timestamp	current_timestamp 结果：2020-05-02 10:41:23

获取日期的年份
月份
日期
	year(日期)
	month(日期)
	day(日期)
		year('2020-05-02')
		结果：2020

日期对应星期几	dayname(日期)	dayname('2020-05-02 10:41:23') 结果：星期六

```
-- 查找1990年出生的学生名单
select 学号,姓名
from student
where year(出生日期)=1990;
```

学号	姓名
0002	猴子
0004	王思聪

23. 查询出1990年出生的学生名单

```
SELECT * from student where YEAR(`出生日期`)='1990'

/*
SELECT * from student where YEAR(`birth`)='1990'
*/
```

24. 查询本月过生日的学生

```
SELECT
  *
FROM
  student
WHERE
  MONTH ( `出生日期` )= MONTH (
  now())
  /*
  SELECT
  *
FROM
  student
WHERE
  MONTH ( `birth` )= MONTH (
  now())
  */
```

25. 查询所有学生的学号、姓名、选课数(别名course_count)、总成绩（别名sum_score）

```
SELECT
  count( score.`课程号` ),
  student.学号,
  student.`姓名`,
  sum(成绩)
FROM
  student INNER JOIN
  score
on
  student.`学号` = score.`学号`
GROUP BY
  student.学号

  /*
  SELECT
  count(score.course_id) as course_count,
  student.id,
  student.name,
  sum(score) as sum_score
FROM
  student INNER JOIN
  score
on
  student.id = score.student_id
GROUP BY
  student.id
  */
```

26. 查询平均成绩大于85的所有学生的学号、姓名和平均成绩(别名avg_score)

```
SELECT
  score.学号,
  student.`姓名`,
  avg(成绩) AS s
FROM
  score,
  student
WHERE
  score.`学号` = student.`学号`
GROUP BY
  学号
HAVING
  s > 85

  /*

  SELECT
  score.student_id,
  student.name,
  avg(score) AS avg_score
FROM
  score,
  student
WHERE
  score.student_id = student.id
GROUP BY
  score.student_id
HAVING
  avg_score > 85

  */
```

27. 查询学生的选课情况：学号(别名 student_id)，姓名(别名 student_name)，课程号(别名 course_id)，课程名称(别名 course_name)

```
SELECT
  st.`学号`,
  st.`姓名`,
  co.`课程号`,
  co.`课程名称`
FROM
  student st
  INNER JOIN score sc ON st.`学号` = sc.`学号`
  INNER JOIN course co ON co.`课程号` = sc.`课程号`

  /*

  SELECT
  st.id as student_id,
  st.name as student_name,
  co.id as course_id,
  co.name as course_name
FROM
  student st
  INNER JOIN score sc ON st.id = sc.student_id
  INNER JOIN course co ON co.id = sc.course_id

  */
```

28. 查询出每门课程的及格人数(别名pass)和不及格人数(别名fail)

```
SELECT
  课程号,
  count( CASE WHEN 成绩 >= 60 THEN '及格' ELSE NULL END ) '及格人数',
  count( CASE WHEN 成绩 < 60 THEN '不及格' ELSE NULL END ) '不及格人数'
FROM
  score
GROUP BY  课程号

/*

SELECT
  course_id,
  count( CASE WHEN score >= 60 THEN 'pass' ELSE NULL END ) 'pass',
  count( CASE WHEN score < 60 THEN 'fail' ELSE NULL END ) 'fail'
FROM
  score
GROUP BY course_id
*/
```

29. 使用分段[100-85],[85-70],[70-60],[<60]来统计各科成绩，分别统计：各分数段人数，课程号和课程名称

```
SELECT
  sc.课程号,
  co.`课程名称`,
  sum( CASE WHEN 成绩 BETWEEN 85 AND 100 THEN 1 ELSE 0 END ) `100-85`,
  sum( CASE WHEN (成绩 < 85 AND 成绩 >= 70 ) THEN 1 ELSE 0 END ) `85-70`,
  sum( CASE WHEN (成绩 < 70 AND 成绩 >= 60 ) THEN 1 ELSE 0 END ) `70-60`,
  sum( CASE WHEN (成绩 < 60 ) THEN 1 ELSE 0 END ) `<60`
FROM
  score sc
  INNER JOIN course co ON sc.`课程号` = co.`课程号`
```

```
GROUP BY
  sc.课程号

  /*
  SELECT
  sc.course_id,
  co.name,
  sum( CASE WHEN score BETWEEN 85 AND 100 THEN 1 ELSE 0 END ) `100-85`,
  sum( CASE WHEN (score < 85 AND score >= 70 ) THEN 1 ELSE 0 END ) `85-70`,
  sum( CASE WHEN (score < 70 AND score >= 60 ) THEN 1 ELSE 0 END ) `70-60`,
  sum( CASE WHEN (score < 60 ) THEN 1 ELSE 0 END ) `<60`
FROM
  score sc
  INNER JOIN course co ON sc.course_id = co.id
GROUP BY
  sc.course_id
  */
```

30. 查询课程编号为0003且课程成绩在80分以上的学生的学号和姓名

```
SELECT
  sc.学号,
  st.`姓名`
FROM
  score sc
  INNER JOIN student st ON sc.`学号` = st.`学号`
WHERE
  课程号 = '0003'
  AND 成绩 > 80

  /*
  SELECT
  sc.student_id,
  st.name
FROM
  score sc
  INNER JOIN student st ON sc.student_id = st.id
WHERE
  course_id = '0003'
  AND score > 80
  */
```

31. 检索"0001"课程分数小于90，按分数降序排列的学生信息

```
SELECT
  sc.学号,
  st.`姓名`
FROM
  score sc
  INNER JOIN student st ON sc.`学号` = st.`学号`
WHERE
  课程号 = '0001'
  AND 成绩 < 90
ORDER BY
  成绩 DESC
  /*
  SELECT
  sc.student_id,
  st.name
FROM
  score sc
  INNER JOIN student st ON sc.student_id = st.id
WHERE
  course_id = '0001'
  AND score <90
ORDER BY
  score DESC
  */
```

32. 查询不同老师所教课程平均分(别名avg_score)从高到低显示

```
SELECT
  avg( sc.成绩 ) 平均分,
  te.`教师姓名`
FROM
  teacher te
  INNER JOIN course co ON co.`教师号` = te.`教师号`
  INNER JOIN score sc ON sc.`课程号` = co.`课程号`
GROUP BY
  te.`教师号`
ORDER BY
  平均分 DESC
  /*
  SELECT
  avg( sc.score ) avg_score,
  te.name
FROM
  teacher te
  INNER JOIN course co ON co.teacher_id = te.id
  INNER JOIN score sc ON sc.course_id = co.id
GROUP BY
  te.id
ORDER BY
  avg_score DESC
  */
```

33. 查询课程名称为"数学"，且分数低于90的学生姓名和分数

```
SELECT
  sc.`成绩`,st.`姓名`
FROM
  score sc INNER JOIN student st on sc.`学号`=st.`学号`
WHERE
  sc.`课程号` IN (
  SELECT
    co.`课程号`
  FROM
    course co
  WHERE
  co.`课程名称` = '数学'
  ) and sc.`成绩` < 90

  /*
  SELECT
  sc.score,st.name
FROM
  score sc INNER JOIN student st on sc.student_id = st.id
WHERE
  sc.course_id IN (
  SELECT
    co.id
  FROM
    course co
  WHERE
  co.name = '数学'
  ) and sc.score < 90
  */
```

34. 查询任何一门课程成绩在70分以上的姓名、课程名称和分数

```
SELECT
  st.`学号`, `姓名`,co.`课程名称`,sc.`成绩`
FROM
  score sc
  INNER JOIN student st ON sc.`学号` = st.`学号`
  INNER JOIN course co ON sc.`课程号` = co.`课程号`
WHERE
  sc.`成绩` > 70



  /*
  SELECT
  st.id, st.name,co.name, sc.score
FROM
  score sc
  INNER JOIN student st ON sc.student_id = st.id
  INNER JOIN course co ON sc.course_id = co.id
WHERE
  sc.score > 70
  */
```

35. 查询不同课程成绩相同的学生的学生编号、课程编号、学生成绩

```
SELECT DISTINCT
  s1.`学号`,
  s1.`课程号`,
  s1.`成绩`
FROM
  score s1
  INNER JOIN score s2 ON s1.`学号` = s2.`学号`
  AND s1.`课程号` != s2.`课程号`
  AND s1.`成绩` = s2.`成绩`
```

36. 查询课程编号为“0002”的课程比“0001”的课程成绩高的所有学生的学号、姓名

```
SELECT
  st.学号,
  st.`姓名`,
  s1.成绩
FROM
  (
    ( SELECT * FROM score WHERE `课程号` = '0001' ) s1
    INNER JOIN ( SELECT * FROM score WHERE `课程号` = '0002' ) s2 ON s1.学号 = s2.学号
    AND s1.成绩 < s2.成绩
  )
  INNER JOIN student st ON s1.学号 = st.`学号`

  /*

  SELECT
  st.id,
  st.name,
  s1.score
FROM
  (
    ( SELECT * FROM score WHERE course_id = '0001' ) s1
    INNER JOIN ( SELECT * FROM score WHERE course_id = '0002' ) s2 ON s1.student_id =
s2.student_id
    AND s1.score < s2.score
  )
  INNER JOIN student st ON s1.student_id = st.id
  */
```

37. 查询学过编号为“0001”的课程并且也学过编号为“0002”的课程的学生的学号、姓名

```
SELECT
  s1.学号,
  st.`姓名`
FROM
  (
    ( SELECT * FROM score WHERE 课程号 = '0001' ) s1
    INNER JOIN ( SELECT * FROM score WHERE 课程号 = '0002' ) s2 ON s1.学号 = s2.学号
  )
  INNER JOIN student st ON s1.学号 = st.`学号`

  /*
  SELECT
  s1.student_id,
  st.name
FROM
  (
    ( SELECT * FROM score WHERE course_id = '0001' ) s1
    INNER JOIN ( SELECT * FROM score WHERE course_id = '0002' ) s2 ON s1.student_id =
s2.student_id
  )
  INNER JOIN student st ON s1.student_id = st.id
  */
```

38. 查询学过“孟扎扎”老师所教的所有课的同学的学号、姓名

```
SELECT
  st.`学号`,st.`姓名`
FROM
  teacher te
  INNER JOIN course co ON co.`教师号` = te.`教师号`
  AND 教师姓名 = '孟扎扎'
  INNER JOIN score sc ON co.`课程号` = sc.`课程号`
  INNER JOIN student st ON st.`学号` = sc.`学号`

  /*
  SELECT
  st.id,st.name
FROM
  teacher te
  INNER JOIN course co ON co.teacher_id = te.id
  AND te.name = '孟扎扎'
  INNER JOIN score sc ON co.id = sc.course_id
  INNER JOIN student st ON st.id = sc.student_id
  */
```

39. 查询没学过"孟扎扎"老师课程的学生姓名

```
SELECT
  *
FROM
  student
WHERE
  学号 NOT IN (
  SELECT
    st.`学号`
  FROM
    teacher te
    INNER JOIN course co ON co.`教师号` = te.`教师号`
    AND 教师姓名 = '孟扎扎'
    INNER JOIN score sc ON co.`课程号` = sc.`课程号`
  INNER JOIN student st ON st.`学号` = sc.`学号`
  )

  /*
  SELECT
  *
FROM
  student
WHERE
  id NOT IN (
  SELECT
    st.id
  FROM
    teacher te
    INNER JOIN course co ON co.teacher_id = te.id
    AND te.name = '孟扎扎'
    INNER JOIN score sc ON co.id = sc.course_id
  INNER JOIN student st ON st.id = sc.student_id
  )
  */
```

40. 查询选修“孟扎扎”老师所授课程的学生中成绩最高的学生姓名及其成绩

```
SELECT
  st.`学号`,
  sc.`成绩`,
  st.`姓名`,
  sc.`课程号`
FROM
  teacher te
  INNER JOIN course co ON co.`教师号` = te.`教师号`
  AND 教师姓名 = '孟扎扎'
  INNER JOIN score sc ON co.`课程号` = sc.`课程号`
  INNER JOIN student st ON st.`学号` = sc.`学号`
ORDER BY
  sc.`成绩` DESC
  LIMIT 1

  /*
  SELECT
  st.id,
  sc.score,
  st.name,
  sc.course_id
FROM
  teacher te
  INNER JOIN course co ON co.teacher_id = te.id
  AND te.name = '孟扎扎'
  INNER JOIN score sc ON co.id = sc.course_id
  INNER JOIN student st ON st.id = sc.student_id
ORDER BY
  sc.score DESC
  LIMIT 1
  */
```

41. 查询至少有一门课与学号为“0001”的学生所学课程相同的学生的学号和姓名

```
SELECT DISTINCT
  st.学号,
  st.姓名
FROM
  score sc
  INNER JOIN student st ON sc.`学号` = st.学号
  AND sc.课程号 IN ( SELECT 课程号 FROM score WHERE 学号 = '0001' )
WHERE
  st.学号 != '0001'

  /*
  SELECT DISTINCT
  st.id,
  st.name
FROM
  score sc
  INNER JOIN student st ON sc.student_id = st.id
  AND sc.course_id IN ( SELECT course_id FROM score WHERE student_id = '0001' )
WHERE
  st.id != '0001'
  */
```

42. 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(别名数学-> math, 语文-> chinese, 英语-> english)

```
SELECT
  学号,
  AVG( 成绩 ),
  MIN( CASE WHEN c.课程名称 = '数学' THEN s.成绩 ELSE NULL END ) AS '数学',
  MIN( CASE WHEN c.课程名称 = '语文' THEN s.成绩 ELSE NULL END ) AS '语文',
  MIN( CASE WHEN c.课程名称 = '英语' THEN s.成绩 ELSE NULL END ) AS '英语'
FROM
  score s
  JOIN course c ON s.课程号 = c.课程号
GROUP BY
  s.学号
ORDER BY
  AVG(
  成绩) desc

  /*
  SELECT
  s.student_id,
  AVG( s.score ) as avg_score,
  MIN( CASE WHEN c.name = '数学' THEN s.score ELSE NULL END ) AS 'math',
  MIN( CASE WHEN c.name = '语文' THEN s.score ELSE NULL END ) AS 'chinese',
  MIN( CASE WHEN c.name = '英语' THEN s.score ELSE NULL END ) AS 'english'
FROM
  score s
  JOIN course c ON s.course_id = c.id
GROUP BY
  s.student_id
ORDER BY
  avg_score desc
  */
```

43. 查询学生平均成绩（别名avg_score）及其名次

```
SELECT
  平均成绩,学号,
  rank() over ( ORDER BY 平均成绩 DESC ) AS ranking
FROM
  ( SELECT avg(成绩) 平均成绩,学号 FROM score GROUP BY 学号 ORDER BY 平均成绩 DESC ) a

  /*
  SELECT
  avg_score,student_id,
  rank() over ( ORDER BY avg_score DESC ) AS ranking
FROM
  ( SELECT avg(score) avg_score,student_id FROM score GROUP BY student_id ORDER BY
avg_score DESC ) a
  */
```

44. 按各科成绩进行排序，并显示排名

```
SELECT
  *,
  rank() over ( PARTITION BY sc.`课程号` ORDER BY sc.`成绩` DESC ) ranking
FROM
  score sc

  /*
  SELECT
  *,
  rank() over ( PARTITION BY sc.course_id ORDER BY sc.score DESC ) ranking
FROM
  score sc
  */
```

45. 查询每门课程成绩最好的前两名学生姓名

```
SELECT DISTINCT
  a.学号,姓名
FROM
  ( SELECT *, rank() over ( PARTITION BY sc.`课程号` ORDER BY sc.`成绩` DESC ) ranking
FROM score sc ) a
  INNER JOIN student st ON a.学号 = st.`学号`
WHERE
  ranking <=2

  /*
  SELECT DISTINCT
  a.student_id,st.name
FROM
```

```
  ( SELECT *, rank() over ( PARTITION BY sc.course_id ORDER BY sc.score DESC ) ranking
FROM score sc ) a
  INNER JOIN student st ON a.student_id = st.id
WHERE
  ranking <=2
  */
```

46. 查询所有课程的成绩第2名到第3名的学生信息及该课程成绩

```
SELECT DISTINCT
  a.学号,姓名
FROM
  ( SELECT *, rank() over ( PARTITION BY sc.`课程号` ORDER BY sc.`成绩` DESC ) ranking
FROM score sc ) a
  INNER JOIN student st ON a.学号 = st.`学号`
WHERE
  ranking > 1
  AND ranking < 4
  /*
  SELECT DISTINCT
  a.student_id,st.name
FROM
  ( SELECT *, rank() over ( PARTITION BY sc.course_id ORDER BY sc.score DESC ) ranking
FROM score sc ) a
  INNER JOIN student st ON a.student_id = st.id
WHERE
  ranking > 1
  AND ranking < 4
  */
```

47. 查询各科成绩前三名的记录

```
SELECT DISTINCT
  a.学号,姓名
FROM
  ( SELECT *, rank() over ( PARTITION BY sc.`课程号` ORDER BY sc.`成绩` DESC ) ranking
FROM score sc ) a
  INNER JOIN student st ON a.学号 = st.`学号`
WHERE
 ranking < 4

 /*
 SELECT DISTINCT
  a.student_id,st.name
FROM
  ( SELECT *, rank() over ( PARTITION BY sc.course_id ORDER BY sc.score DESC ) ranking
FROM score sc ) a
```

```
  INNER JOIN student st ON a.student_id = st.id
WHERE
 ranking < 4
 */
```