数据库系统和信息管理期末复习

关系代数 & SQL 语句题

2025 年 6 月考试该题分值为 40 分

大概可能问到以下几个问题：
1. 画出对应的 E-R 图
2. 写出上述实体和关系的关系模式（Relational Schema），标明主码（下划线）和外键（并注明对应的引用）
3. 联系的主码的选择
4. SQL 代码书写（见 SQL 专题文档）

1.

个人说明 Gother
个人喜欢用连接（特别是等值连接，一般不用自然连接），但是有的时候需要用到笛卡尔积的时候，一定要注意要σ查询一下，保证 id 之类的相等

$$\Pi_{id,name}\sigma_{project.teacher\_id=teacher.id}(\sigma_{dept\_name='software'}(teacher)\times project)$$

如果使用连接的话就不用写σ了

$$\Pi_{id,\ name}\ (\sigma_{dept\_name='software'}(teacher\bowtie_{teacher.id=progect.teacher\_id}project\ )$$

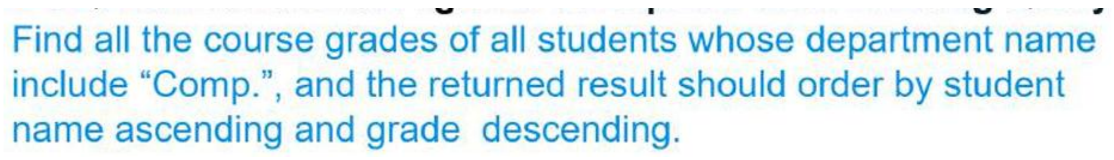
再比如：

$\Pi_{id,name}\sigma_{student\_id=student.id}(\Pi_{student\_id,project\_id}(participate)\div\Pi_{project\_id}\sigma_{student\_id='12345'}(participate)\times student)$

可以写成

$\Pi_{id,name}(student\bowtie(\Pi_{student\_id,project\_id}(participate)\div\Pi_{project\_id}\sigma_{student\_id='12345'}(participate)))$

其次，不推荐用 except，因为拼不对英语单词

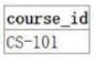

Find all the course grades of all students whose department name include “Comp.”, and the returned result should order by student name ascending and grade descending.

```
Select S.*,T.*
From student as S , takes as T
Where S.ID=T.ID  and dept_name like  '%Comp.%'
Order by S.name, grade desc
```

$$\Pi_{S.*,T.*}\left(\sigma_{dept\_name\ like\ '\%Comp.\%' \wedge S.ID=T.ID}\left(\rho_S(student) \times \rho_T(takes)\right)\right)$$

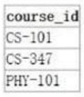
Ex1: “Find all courses opened in both in the Fall 2017 semester and in the Spring 2018 semester”

//先前用“集合交”查询实

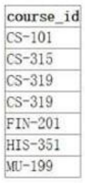


course_id
CS-101

```
select course_id   from section  where semester = 'Fall' and year = 2017
intersect
select course_id  from section  where semester = 'Spring' and year= 201
```

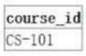


course_id
CS-101
CS-347
PHY-101



course_id
CS-101
CS-315
CS-319
CS-319
FIN-201
HIS-351
MU-199

//这里用“空关系测试”嵌套查询实现

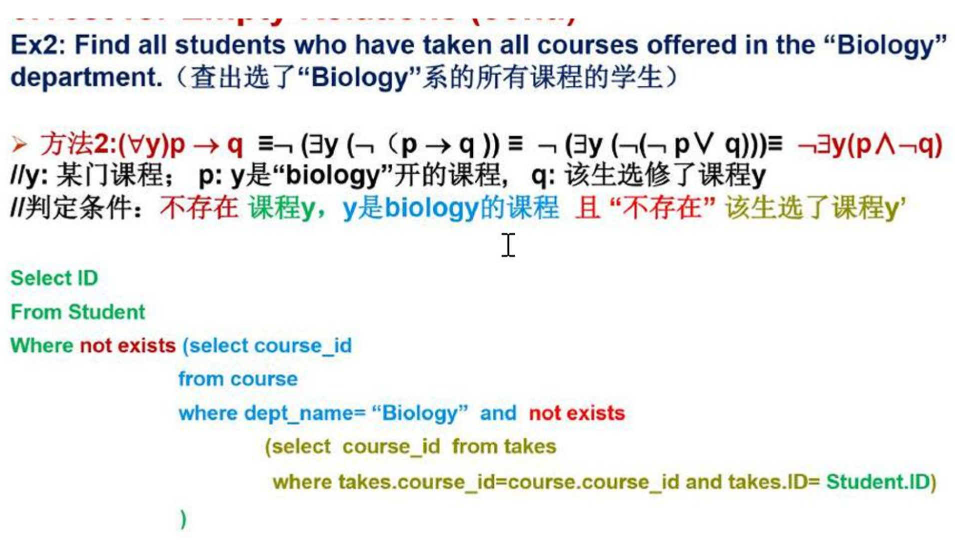


course_id
CS-101

```
  select course_id
  from section as S
  where semester = 'Fall' and year = 2017 and exists
                    (select couse_id  from section as T
                    where semester = 'Spring' and year= 2018
                           and S.course_id = T.course_id);
```

$\boxed{S.course\_id = T.course\_id)}$

//解析：“在父查询结果中存在的course_id，在子查询结果中也存在”



Ex2: Find all students who have taken all courses offered in the "Biology" department.（查出选了"Biology"系的所有课程的学生）

➢ 方法2:$(\forall y)p \rightarrow q \equiv \neg(\exists y(\neg(p \rightarrow q))) \equiv \neg(\exists y(\neg(\neg p \vee q))) \equiv \neg\exists y(p \wedge \neg q)$

//y: 某门课程；p: y是"biology"开的课程，q: 该生选修了课程y

//判定条件：不存在 课程y，y是biology的课程 且 "不存在" 该生选了课程y'

```
Select ID
From Student
Where not exists (select course_id
                  from course
                  where dept_name= "Biology"  and  not exists
                        (select  course_id  from takes
                         where takes.course_id=course.course_id and takes.ID= Student.ID)
                  )
```

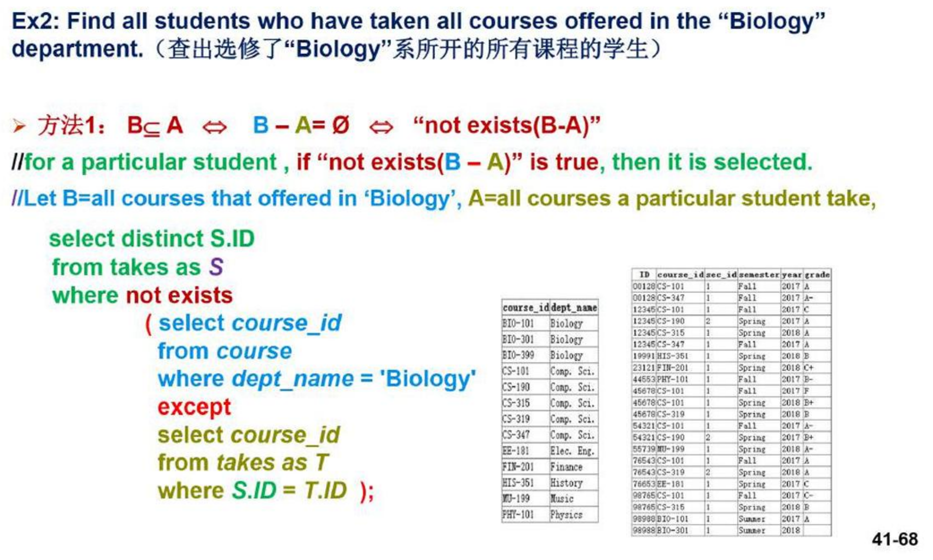
用离散数学的观点解答问题。

问题翻译成：对于任意生物专业开设的课程，学生就一定选了（对于任何课程，只要是生物学院开设的，学生就选了）

通过等价变换，得到：不存在这样的课程，课程是生物学院开设的，但是学生没有选

进一步向数据库方面的语言转化：不存在这样的课程，课程在生物学院开设的课程中，但是课程不在学生选课的表格中。

这样就转化成了两个不在和一个在，不在我们用 not exists。在对学生的表的查询中添加约束。



Ex2: Find all students who have taken all courses offered in the "Biology" department.（查出选修了"Biology"系所开的所有课程的学生）

➢ 方法1：$B \subseteq A \Leftrightarrow B - A = \emptyset \Leftrightarrow$ "not exists(B-A)"

//for a particular student , if "not exists(B – A)" is true, then it is selected.

//Let B=all courses that offered in 'Biology', A=all courses a particular student take,

```
select distinct S.ID
from takes as S
where not exists
        ( select course_id
          from course
          where dept_name = 'Biology'
          except
          select course_id
          from takes as T
          where S.ID = T.ID  );
```

course_id	dept_name
BIO-101	Biology
BIO-301	Biology
BIO-399	Biology
CS-101	Comp. Sci.
CS-190	Comp. Sci.
CS-315	Comp. Sci.
CS-319	Comp. Sci.
CS-347	Comp. Sci.
EE-181	Elec. Eng.
FIN-201	Finance
HIS-351	History
MU-199	Music
PHY-101	Physics

ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	

41-68

同时，在这里给出另一种解答方法

学生选的课程一定包含生物学院开设的课程（也就是说，生物学院开设的课程是学生选择的课程的子集），假设学生选择的课程集合是 A，生物学院开设的课程的几何是 B，那么 B⊆A，也就是说 B-A=φ空集，也就是说不存在 B-A，not exists(B-A)，减号用 except 体现

在 A 的查询中添加约束

1. 2022-2023 期末考试题

customer(ID, customer_name)
account(account_number, ID, balance)
loan(loan_number, ID, branch_name, amount)
branch(branch_name)

Section I: Relational Algebra (1)–(4)

(1) Find all the bank depositing account_numbers of the customer whose name is ’张三’.
(2) Find the customer IDs and names who has not borrow any loan from bank.
(3) List ID and the summary of all account balances of every customer.
(4) List IDs of the customers who have borrowed loans from all branches.

Section II: SQL (5)–(10)

(5) Query the account_numbers of customers whose names include “君”.
(6) Query the IDs of the customers who have at least one loan but no any account.
(7) List each bank branch_names and the quantity of customers who have loans from the branch, and output the tuples in descending order of the quantity.
(8) Query IDs of the customers whose balance summary of his/her depositing accounts is greater than 10000.
(9) Query IDs of the customers whose balance summary of his/her depositing accounts is the largest.
(10) Find the customer IDs who have borrowed loans from all the bank branches which the customer (ID=“A101”) has borrowed loans from.

文字备用版本:

customer(ID, customer_name)
account(account_number, ID, balance)
loan(loan_number, ID, branch_name, amount)
branch(branch_name)

Section I: Relational Algebra (1)–(4)

(1) Find all the bank depositing account_numbers of the customer whose name is ’张三’.
(2) Find the customer IDs and names who has not borrow any loan from bank.
(3) List ID and the summary of all account balances of every customer.
(4) List IDs of the customers who have borrowed loans from all branches.

Section II: SQL (5)–(10)

(5) Query the account_numbers of customers whose names include “君”.
(6) Query the IDs of the customers who have at least one loan but no any account.
(7) List each bank branch_names and the quantity of customers who have loans from the branch, and output the tuples in descending order of the quantity.
(8) Query IDs of the customers whose balance summary of his/her depositing accounts is greater than 10000.
(9) Query IDs of the customers whose balance summary of his/her depositing accounts is the largest.
(10) Find the customer IDs who have borrowed loans from all the bank branches which the customer (ID=“A101”) has borrowed loans from.

Please use relational algebra language to express query requirements (1)- (4):

(1) Find all the bank depositing account_numbers of the customer whose name is '张三'

$\Pi_{account\_number}$ ($\sigma_{customer\_name=\text{'张三'}}$ (customer ⋈ account))

Or

$\Pi_{account\_number}$ ($\sigma_{customer\_name=\text{'张三'} \text{ and } customere.ID=account.ID}$(customer × account))

(2) Find the customer IDs and names who has not borrow any loan from bank.

$\Pi_{ID,\ customer\_name}$ (customer) - $\Pi_{ID,\ customer\_name}$ ($\sigma_{customer.ID=loan.ID}$ (customer × loan))

Or

$\Pi_{ID,\ customer\_name}$ (customer) - $\Pi_{ID,\ customer\_name}$ (customer ⋈ loan)

(3) List ID and the summary of all account balances of every customer.

$\Pi_{ID,\ s1}$ ($_{ID}\mathcal{G}_{sum(balance)\ as\ s1}$ ($\sigma_{account.ID=customer.ID}$(account × customer)))

Or $_{ID}\gamma_{sum(balance)\ as\ s1}$ ($\sigma_{account.ID=customer.ID}$(account × customer))

Or $_{ID}\gamma_{sum(balance)\ as\ s1}$ ($\sigma_{account.ID=customer.ID}$(account))

(4) List IDs of the customers who have borrowed loans from all branches.

$\Pi_{ID,branch\_name}$ (loan) ÷ Π_{branch_name} (branch)

or

$\Pi_{ID,branch\_name}$ (loan ⋈ branch) ÷ Π_{branch_name} (branch)

or

$\Pi_{ID,branch\_name}$ (loan ⋈ customer) ÷ Π_{branch_name} (branch)

Please useSQL language to express query requirements (5)-(10):

(5) Query the account_numbers of customers whose names include "君";

```
SELECT account_number
  FROM loan natural join customer
  WHERE customer_name like '%君%';
```

(6) Query the IDs of the customers who have at least one loan but no any account.

```
SELECT distinct ID
FROM  loan
WHERE  ID  not in
              ( SELECT ID
                FROM  account);
```

Or

```
SELECT distinct ID
FROM  loan
except
select distinct ID
FROM account;
```

Or

```
SELECT ID
FROM loan
Group by loan.ID
HAVING count(loan_number)>=1
Except
Select distinct ID
From account;
```

(7) List each bank branch_names and the quantity of customers who have loans from the branch, and output the tuples in descending order of the quantity;

```
SELECT   branch_name, count(distinct ID) as quantity
FROM   loan
GROUP BY branch_name
ORDER BY quantity desc;
```

(8) Query IDs of the customers whose balance summary of his/her depositing accounts is greater than 10000;

```
SELECT   ID
FROM   account
GROUP BY ID
HAVING sum(balance) > 10000;
```

(9) Query IDs of the customers whose balance summary of his/her depositing accounts is the largest;

```
SELECT   ID
FROM   account
GROUP BY ID
HAVING sum(balance)>=all (SELECT sum(balance)
                          FROM   account
                          GROUP BY ID);
```

(10) Find the customer IDs who have borrowed loans from all the bank branches which the customer (ID= "A101") has borrowed loans from;

方法 1： Let X: all the bank branches which customer ID= "A101" has borrowed loans from
Let Y: all the bank branches which the specular customer who has borrowed loans from
Then : SELECT ID FROM loan WHERE not exist(X-Y);

```
SELECT ID
FROM loan A
WHERE not exists (
        (select distinct branch_name
         from loan
        where ID ='A101')
      except
       (select distinct branch_name
       from loan
        where ID =A.ID)   );
```

方法 2：选出某客户的条件是： 不存在某支行，客户 A101 贷款了，而该客户没有贷款记录。

```
Select ID
From customer
Where not exists
        (select * from loan L1
         Where ID='A101' and not exists
                              (select * from loan L2
                               Where L2.branch_name=L1.branch_name
                               And L2.ID=customer.ID)
```

2. From 练习题

Consider the relational database of a university research project with the following relation schemas, where the primary keys are underlined.

teacher(<u>ID,</u> name, dept_name)

project(<u>ID</u>, name, teacher_ID,budget)

student(<u>ID</u>, name, age, dept_name)

participate<u>(student_ID,project_ID,</u> salary)

Note: Each project has a teacher as a leader, represented by attribute teacher_ID.

1、 Give a relational algebra expression for each of the following queries:

(1) List the Ids and names of all projects whose leader is the teacher of the "Software" department.

(2)List the Ids and names of all members who have participated in the project named " AI" .

(3) List the IDs of all students who have taken more than five projects

(4) List the IDs and names of all students who have participated all projects that the student with ID "12345" has participated.

2、 Write SQL statements to perform the following commands:

(1)List Ids and names of all projects with a budget greater than 10,000 and with a name include "Software".

(2)List Ids and names of all student of "Software" department who have NOT participated in any project.

(3)List Ids and names of all projects and the number of students who have participated in it .Sort the results in descending order based on the number of students.

(4)List Ids of projects whose students earn a higher salary on average than the average salary at project with id '001'.

(5) List the names, sum salary of all students who have the most (highest) sum salary.

(6)List the names of all students who have participated in all projects led by the teacher whose name is 'Enstein';.

1、 Give a relational algebra expression for each of the following queries:

“关系代数”语言表达查询：

select: project: union: set difference: –, Cartesian product: x, rename

选用最基本的操作：并，差，选择，投影，笛卡尔积（rename：一般不涉及）。

注意：集合“并”可以由 “交”和“差”表示出来!

注意：一般增加了考“聚集”操作和“除法”操作（课件里有讲）

(1) List the Ids and names of all projects whose leader is the teacher of the “Software” department.

列出所有项目中其指导教师是“软件“系教师的指导老师的标识 ID 和姓名 name

分析：输出：教师的 ID, name，条件：有指导项目的“软件”系的教师。输入：教师和项目表。

最简单思路：就是作连接后进行筛选，最后投影。

注意： 不能用自然连接，因为 project 和 teacher 相同的 ID 不是同一属性!

$$\Pi_{\text{id,name}}\sigma_{\text{project.teacher\_id=teacher.id}}(\sigma_{\text{dept\_name='software'}}(\text{teacher}) \times \text{project})$$

$$\Pi_{\text{teacher.id,name}}\sigma_{\text{project.teacher\_id=teacher.id}}(\sigma_{\text{dept\_name='software'}}(\text{teacher})\times \text{project})$$

(2)List the Ids and names of all members who have participated in the project named “ AI” .

列出参与了项目名为’AI’的所有学生的 ID 和 name

分析：输出表是 student. 筛选条件设计项目名在 project 里，通过 participate 才能关联起来，所以需要连接，筛选和投影。

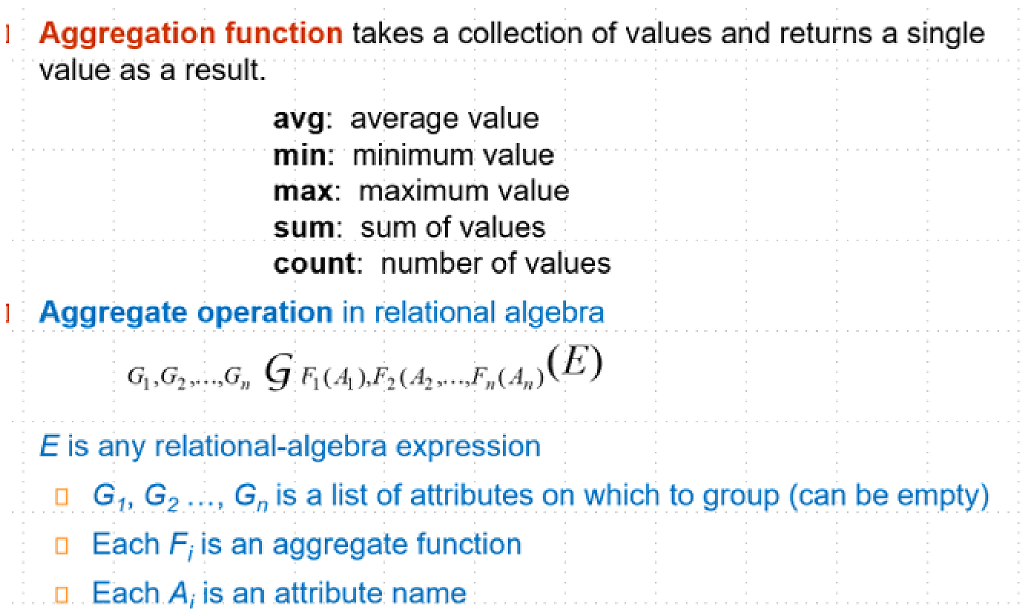
$$\Pi_{\text{id,name}}\sigma_{\text{project.name='AI'}\land\text{student.id=participate.studentd\_id}}$$
$$(\sigma_{\text{project.id=participate.project\_id}}(\text{project}) \times(\text{participate})) \times(\text{student}))$$

$$\Pi_{\text{id,name}}\sigma_{\text{student.id=participate.studentd\_id}}$$
$$(\sigma_{\text{project.id=participate.project\_id}}(\sigma_{\text{project.name='AI'}}(\text{project}))\times\text{participate}) \times(\text{student}))$$

(3) List the IDs of all students who have taken more than five projects

列出参与了 5 个项目以上的学生的 ID

分析：首先判定有聚集操作（超出本科版，课件里有讲 **Extended Relational-Algebra-Operations**）



- **Aggregation function** takes a collection of values and returns a single value as a result.

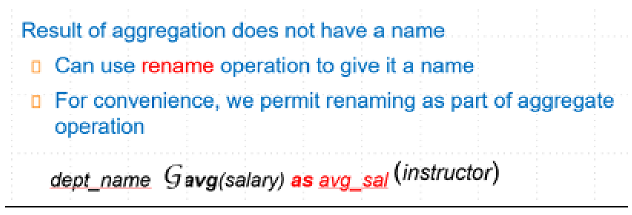
  avg: average value
  min: minimum value
  max: maximum value
  sum: sum of values
  count: number of values

- **Aggregate operation** in relational algebra

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2,\ldots,F_n(A_n)}(E)$$

E is any relational-algebra expression

- $G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name



Result of aggregation does not have a name

- Can use rename operation to give it a name
- For convenience, we permit renaming as part of aggregate operation

$$_{dept\_name}\mathcal{G}_{\textbf{avg}(salary)\ \textbf{as}\ avg\_sal}(instructor)$$

$$\Pi_{student\_id}\ \sigma_{number\_project>5}\ ({}_{student}\mathcal{G}_{count\ distinct(student\_id)\ as\ num\_project}(participate)$$

(4) List the IDs and names of all students who have participated all projects that the student with ID “12345” has participated.

列出参与了学生 ID='12345'的学生所参与的全部项目的学生的 ID，name，
分析：关系代数表达形如若查询诸如“选修了全部课程”的学生、“使用了全部零件”的工程等，需用除法操作实现！（参见 LEC8 CHAPTER6 的 PPT 中 **Additional Operations*）！**

$$\Pi_{id,name}\sigma_{student\_id=student.id}$$
$$(\Pi_{student\_id,project\_id}(participate)\div\Pi_{project\_id}\sigma_{student\_id='12345'}(participate)\times student)$$

$$\Pi_{id,name}\ \sigma_{student\_id=student.id}$$
$$(\Pi_{student\_id,project\_id}(participate)\div\Pi_{project\_id}\sigma_{student\_id='12345'}(participate))$$

2、 Write SQL statements to perform the following commands:

"SQL"语言表达查询：

注意： 重点是考察初级 SQL 的灵活使用

条件语句：like 字符串匹配"%"，"-"， between…and：数值范围含等号。

嵌套查询：融合集合运算，多表连接，聚集函数等。

(1)List Ids and names of all projects with a budget greater than 10,000 and with a name include "Software".

列出项目预算大于 10000 的并且项目名包含"软件"的项目 ID，name

分析：条件在 project 中都可以找到字段，所以输入只 project，考察 where 条件写法！

```
select id,name from project where budget>10000 and name like ' %Software %';
```

(2)List Ids and names of all student of "Software" department who have NOT participated in any project.

列出"software"系没有参与项目的学生 ID 和 name

分析：输出取自 student，筛选条件是"没有参与项目"涉及 participate，并且是没有参与。如果直接连接 student 和 participate 得到的是"参与项目"的，所以应该嵌套子查询~集合成员资格进行筛选。

```
select id,name from student
where dept_name='Software' and id not in
                              (select distinct student_ID from participate );
```

若采用"集合差"运算：注意：except 和 except all 的区别：

```
 (select id,name from student
  where dept_name='Software')
except
  (select id,name
  from student, participate
  where id=student_ID);
```

(3)List Ids and names of all projects and the number of students who have participated in it .Sort the results in descending order based on the number of students.

列出项目的 ID,name 和参与该项目的学生数。并且输出结果按学生数降序排列。

分析：涉及聚集函数因为有统计学生数，主要涉及 pariticipate 表，分组条件是 project_ID。考虑输出字段在 project 表，所以需要和 student 表作连接。考虑到输出结果"学生数"需要作为排序条件，所以需要取别名以方便表达。(注意不要写成自然连接，因为相同属性在各表中名称不同)

```
select id,name,count(*) as stu_nums
from participate,project
where participate.project_id=project.id group by id order by stu_nums desc;
```

(4)List Ids of projects whose students earn a higher salary on average than the average salary at project with id '001'.

列出比项目ID='001'的平均salary高的项目ID

分析：涉及聚集函数，并且包含having条件筛选。

```
select  project_id  from  participate  group by project_id
having avg (salary) >(select avg (salary) from participate where project_id= '001')
```

(5) List the names, sum salary of all students who have the most (highest) sum salary.

列出所有获得项目最高sum(salary)的学生的姓名及其sum(salary)

分析：本类题涉及聚集函数的，嵌套子查询中的**集合比较**!

聚集函数的筛选条件是：输出学生的sum(salary) >=all（所有学生的sum(salary)）。

```
select student.name,sum(salary)
from student, participate
where student.id= participate.student_id
group by student.id
having sum(salary)>= all
    (select sum(salary)
    from student, participate
    where student.id= participate.student_id
    group by student.id );
```

```
select student.name,sum(salary)
from student, participate
where student.id= participate.student_id
group by student.id
having sum(salary)>= all
    (select sum(salary)
    from participate
    group by student.id );
```

(6)List the names of all students who have participated in all projects led by the teacher whose name is 'Enstein';.

列出参与了由教师"Enstein"所指导的**所有项目**的学生姓名 name

分析：类似于关系代数除法，SQL 常采用**嵌套子查询的空关系测试（exists 或 not exists）实现**（注意：因为没有数量，不能用集合比较（>=all））该查询转换为"不存在一个项目是"Enstein"指导的，而该学生没有选！"

注意：需要采用"相关子查询"，参考 PPT LEC4 教材 3.8.3 空关系测试

```
    select name
     from student s
    where not exists
    (
      select project.id from teacher,project

  where   teacher.id=project.teacher_id and teacher.name='Enstein'
  except
  select participate.project_id
  from student, participate
  where student.id=participate.student_id and student.id=s.id;
)
```

3. 陈鹏练习题

4.

有如下数据库模式（下划线标记的是主码）.

user (ID, name, team, phone) //用户（ID，姓名，小组，电话）

dev (dNo, dName, type)　　//设备（设备编号，名称，类型）

adm (NO, name, salary, store)　　//库房管理员（编号，姓名，薪水，库房）

borrow (ID, dNo, date, NO)　　//设备借用（用户 ID，设备编号，日期，经手库管编号）

1、请使用关系代数完成以下查询:

(1) 列出小组为“锦江”的用户的 ID 和姓名

(2) 列出名为“李四”的用户借用过的设备的编号和名称'

(3) 列出 ID 为 777 的用户借用过且经手库管编号为 999 的设备的编号

(4) 列出满足如下条件的用户的 ID 和姓名：此用户借用了 ID 为 888 的用户借用过的所有设备

2、通过 SQL 完成如下查询:

(1) 列出薪水高于 5000 的管理员姓名.

(2) 列出管理员平均薪水高于 5000 的库房

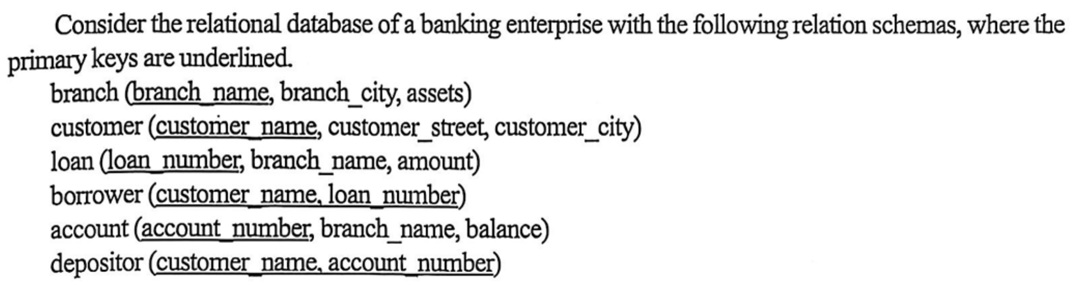
(3) 列出在各自所属库房中薪水最高的管理员的姓名.

(4) 列出在'2022-6-1'和''2022-6-30'之间被 ID 为 666 的用户借用过的设备编号和名称

(5) 列出从来没有由编号为 555 的库管经手借用过设备的用户的姓名

(6) 列出满足如下条件的用户 ID 和姓名：该用户借用过所有曾经被 555 号管理员经手借出过的设备。

5.



Consider the relational database of a banking enterprise with the following relation schemas, where the primary keys are underlined.

branch (branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
loan (loan_number, branch_name, amount)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)

1. Give a relational algebra expression for each of the following queries:

(1) Find all loan_number for loans made at the 'Perryridge' branch (that is the name of a certain branch) with loan amounts greater than $1000.

(2) Find names of all customers who have accounts in all branches located in 'Brooklyn'.

(3) Find the max account balance in branch located in 'Brooklyn'.

2. Write SQL statements to perform the following commands:

(1) Find the name of all customers who have accounts but not loans.

(2) Find the name of all customers who have accounts in all branches located in 'Brooklyn'.

(3) Find the name of all branches that have assets greater than those of at least one branch located in 'Brooklyn'.

(4) Find the names of all branches where the average account balance is more than $2000.

(5) Find all customers that live in the same city with branches they open accounts.

1、Give a relational algebra expression for each of the following queries:

(1) Find all loan number for loans made at the 'Perryridge' branch (that is the name of a certain branch) with loan amounts greater than $1000

$\Pi_{\text{loan\_number}}\ \sigma_{\text{branch\_name='Perryridge branch' and amount>1000}}(\text{loan})$ ✓

(2)Find names of all customers who have accounts in **all** branches located in 'Brooklyn'.

Π ~~customer_name~~

(3) Find the max account balance in branch located in 'Brooklyn'

$\text{branch\_name}\ \mathcal{G}\ \text{max(balance)}\ (\sigma_{\text{cbranch\_ity='Brooklyn'}}(\text{account} \bowtie \text{Branch}))$

2、Write SQL statements to perform the following commands:

(1)Find the name of all customers who have accounts but not loans.
(Select customer_name from depositor) except (Select customer_name from borrower)

(2)Find the name of all customers who have accounts in all branches located in 'Brooklyn'.

(3)Find the name of all branches that have assets greater than those of at least one branch located in 'Brooklyn'.
Select branch_name from branch where assets>some(select assets from branch where branch_city='Brooklyn'

(4)Find the names of all branches where the average account balance is more than $2000.
Select branch_name from account group by branch_name having avg(balance)>2000

(5)Find all customers that live in the same city with branches they open accounts.
Select customer_name, customer_city from ((customer natural join depositor) natural join account) natural join branch where customer_city= branch_city

6.

II. Queries. (5 points each; 40 total)

An employee training database contains following relational schemas (primary keys are underlined):
employee(<u>ID</u>, name, address, salary)
course(<u>course_id</u>, title, hours)
section(<u>course_id, sec_id</u>, start_date, finish_date)
takes(<u>employee_id, course_id, sec_id</u>, grade)

1. Give a relational algebra expression for each of the following queries:
(1) List the IDs and names of all employees who have taken a course titled “Company Culture”.
(2) List the IDs and names of all employees who have taken all courses that the employee with ID “dev101” has taken.
(3) List the IDs of all employees who have taken more than five courses.

2. Write SQL statements to perform the following commands:
(1) List the IDs and names of all employees who have NEVER taken the course titled “Business Decision”.
(2) List the IDs and titles of all courses whose title begin with “Business”.
(3) List the ID and name of the employee who has taken the course titled “Business Decision” and got the highest grade.
(4) List the IDs and names of all employees who have taken all courses that the employee with ID “dev101” has taken.
(5) List the employee names and the average grades of courses they have taken.

1. **Give a relational algebra expression for each of the following queries:** 注意可能有多种解法

(1) List the IDs and names of all employees who have taken a course titled “Company Culture”.

$\Pi_{\text{employee.ID,name}}\sigma_{\text{employee.id=takes.employee\_ID} \wedge \text{takes.course\_i=course.course\_id} \wedge \text{title='Company Culture'}}(\text{employee} \times \text{takes} \times \text{section})$

(2) List the IDs and names of all employees who have taken all courses that the employee with ID “dev101” has taken.
$\Pi_{\text{employee.ID,name}}\sigma_{\text{employee.ID=takes.employee\_id}}(\Pi_{\text{employee\_id,course\_id}}(\text{takes}) \div \Pi_{\text{course\_id}}\sigma_{\text{employee\_id='dev101'}}(\text{takes}) \times \text{employee})$

(3) List the IDs of all employees who have taken more than five courses.

$\Pi_{\text{employee\_id}}\sigma_{\text{number\_course>5}}(\text{employee\_id}\ \mathcal{G}\ \text{count-distinct(course\_id) as num\_course}\ (\text{takes})$

2. **Write SQL statements to perform the following commands:**

(1) List the IDs and names of all employees who have NEVER taken the course titled "Business Decision".

```
select id, name
  from employee
  where ID not in
  (select employee_id
     from takes, course
     where takes.course_id = course.course_id and title=' Business Decision');
```

(2) List the IDs and titles of all courses whose title begin with "Business".

```
select couse_id, title
  from course
  where title like 'Business%';
```

(3) List the ID and name of the employee who have taken the course titled "Business Decision" and get the highest grade.

```
select id, name
  from employee e, takes t, course c
  where ID=t.employee_id and t.course_id = c.course_id title=' Business Decision' and grade =
  (select max(grade)
     from takes t, course c
     where t.course_id=c.course_id and title=' Business Decision');
```

(4) List the IDs and names of all employees who have taken all courses that the employee with id "dev101" has taken.

```
select ID, name
  from employee e
  where not exists
  ((select course_id
     from takes
     where employee_id='dev101')
   except
   (select course_id
     from takes
     where e.ID=employee_id));
```

(5) List the employee names and the average grades of courses they have taken.

```
select name, avg(grade)
  from employee, takes
  where ID=employee_id
  group by ID, name;
```

7.

The following relational schemas store information about students and research projects that they have taken part in. (The primary keys are underlined):
student (ID, name, dept_name)
project(project_no, title)
s_p(student_id, project_no, grade)

1. **Give a relational algebra expression for each of the following queries:**
   (1) List the IDs and names of all students who have taken part in the project titled "Wechat Programming".
   (2) List the IDs and names of all students who have taken all projects that the student with ID "CS101" has taken part in.
   (3) List the IDs of all students who have taken part in more than five projects.

2. **Write SQL statements to perform the following commands:**
   (1) List the IDs and names of all students who have **NEVER** taken part in the project titled "Wechat Programming".
   (2) List the project numbers and titles of all projects whose title begin with "Program".
   (3) List the ID and name of the employee who have taken part in the project titled "Programming of Wechat" and get the highest grade.
   (4) List the IDs and names of all students who have taken part in all projects that the student with ID "CS101" has taken part in.
   (4) List the students' names and the average grades of projects they have taken part in.

//

1. A car insurance company stores important data about customers, vehicles, insurance policies, payments, and accidents. The following entity sets are included in the system:

Each **customer** has a unique customer_id, along with attributes such as name and address. Each **car** is identified by a license_no, and is associated with a model and a customer_id who owns it. Each **insurance policy** is uniquely identified by policy_id. A policy may have multiple payments. Each **payment** has a payment_no, due_date, amount, and received_on, and is associated with a specific policy_id. This records **accident reports**, each identified by report_id, with attributes date and place. This relationship indicates which cars were involved in which accidents, linking license_no and report_id. This indicates which policies cover which cars, linking license_no and policy_id.

Note:

- A customer can own multiple cars.
- A car can be involved in multiple accidents and be covered by multiple policies.
- A policy can have multiple payments.
- A car may participate in multiple accidents, each with a corresponding report.

Write the relational schema for each entity and relationship, underlining primary keys and clearly marking foreign key references. (10 points)

2. A package delivery system tracks the sending and receiving of packages between customers. Each **Customer** has a unique customer_id, and attributes like name, address, and telephone. Each **Packet** has a unique packet_id, and includes attributes like weight and expense.

Customers **send** and **receive** packets. The **Send** relationship tracks which customer sends which packet and includes the attribute time_send. The **Receive** relationship records which customer receives which packet and includes the attribute time_receive.

Packets are transported **through** various **Places**, each identified by place_id, with attributes place_name, city, and country.

Draw the ER diagram based on the description above. Use appropriate cardinalities and attributes. (10 points)

To build a database to handle information about a college. Each student can take multiple courses. Each course can be taught by multiple teachers. Each teacher can teach more than one course. Each student can be guided by a teacher. Each teacher guides more than one student.

1. Construct an E-R diagram that captures the information above.
2. Convert the E-R diagram to 3NF relations. Specify keys and referential integrity constraints.
3. In the relationship mode you designed, write out the SQL statement to complete the following query:

Ask for the No and Name of all students who choose "市场营销" teached by "王敏".