CS 6016

Database Systems

Adv. Queries II

This week

- Advanced Queries

- Teamwork
  - Project Phase 2 / Phase 3 / Phase 4

Ponder

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
4	999-9999
5	777-7777

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- Find all people with the same phone number

Phones

CardNum	Phone
1	555-5555
2	666-6666
2	555-5555
3	777-7777
4	888-8888
5	777-7777

Phones × Phones

CardNum	Phone	CardNum	Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
1	555-5555	5	777-7777
2	666-6666	1	555-5555
2	666-6666	2	666-6666
2	666-6666	2	555-5555
2	666-6666	3	777-7777
2	666-6666	4	888-8888
…	…	…	…

Self-Join

- First we have to disambiguate

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 $\times$ P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

Self-Join

- Then filter by matching phone number

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 $\times$ P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

- $\sigma_{P1.Phone=P2.Phone}(P1 \times P2)$

Self-Join

- Then filter by *not* matching card number

- ρ(P1, Phones)
- ρ(P2, Phones)

P1 $\times$ P2

P1.CardNum	P1.Phone	P2.CardNum	P2.Phone
1	555-5555	1	555-5555
1	555-5555	2	666-6666
1	555-5555	2	555-5555
1	555-5555	3	777-7777
1	555-5555	4	888-8888
…	…	…	…

- $\sigma_{\text{P1.Phone=P2.Phone \&\& P1.CardNum != P2.CardNum}}(\text{P1} \times \text{P2})$

SQL Self-Join

```
select p1.CardNum, p2.CardNum
from Phones p1 join Phones p2
where p1.Phone = p2.Phone
and p1.CardNum != p2.CardNum;
```

- Renaming (aka “range variables”) required for self-join

Join

- Default `join` is called an inner join
- `x JOIN y WHERE ...`
  - Gives rows where condition is true
- Equivalent:
  - `x INNER JOIN y`
  - `x, y`

Outer Join

- Two types of outer join: `LEFT` and `RIGHT`
- `ON` clause required!
- `x LEFT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `x`

Outer Join

- Two types of outer join: `LEFT` and `RIGHT`
- `ON` clause required!
- `x LEFT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `x`
- `x RIGHT JOIN y ON condition`
  - Gives all rows where condition is true
  - And gives all rows from `y`

Left Join

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

Left Join

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

```
Patrons p LEFT JOIN CheckedOut c
ON p.CardNum = c.CardNum;
```

Name	CardNum	CardNum	Serial
Joe	1	1	1001
Joe	1	1	1004
Ann	2	NULL	NULL
Ben	3	NULL	NULL
Dan	4	4	1005

Left Join

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005

```
Patrons p LEFT JOIN CheckedOut c
ON p.CardNum = c.CardNum;
```

Name	CardNum	CardNum	Serial
Joe	1	1	1001
Joe	1	1	1004
Ann	2	NULL	NULL
Ben	3	NULL	NULL
Dan	4	4	1005

- Unmatched tuples get `NULL` in right-side columns

Shortcut

```
Patrons NATURAL LEFT JOIN CheckedOut;
```

Name	CardNum	Serial
Joe	1	1001
Joe	1	1004
Ann	2	NULL
Ben	3	NULL
Dan	4	1005

Shortcut

```
Patrons NATURAL LEFT JOIN CheckedOut;
```

Name	CardNum	Serial
Joe	1	1001
Joe	1	1004
Ann	2	NULL
Ben	3	NULL
Dan	4	1005

Only one copy of natural column(s)

NULL

- `NULL` is a special value in SQL
  - It does not have the reflexive property
  - i.e. `NULL != NULL`

NULL

- `NULL` is a special value in SQL
  - It does not have the reflexive property
  - i.e. `NULL != NULL`
- `WHERE CardNum != NULL` `// wrong`
- `WHERE CardNum IS NOT NULL` `// right`

NULL

- Boolean operators on `NULL` always return `NULL`

  - `NULL == 5` → `NULL`
  - `NULL != 5` → `NULL`
  - `NULL == NULL` → `NULL`
  - `NULL != NULL` → `NULL`

NULL

- Boolean operators on `NULL` always return `NULL`
  - `NULL == 5` → `NULL`
  - `NULL != 5` → `NULL`
  - `NULL == NULL` → `NULL`
  - `NULL != NULL` → `NULL`
- If `NULL` is used where a Boolean is expected, interpreted as `FALSE`

Quiz

- Find Names of Patrons who have not checked out a book. Use outer join.

CheckedOut

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Patrons

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Solution

- Find Names of Patrons who have not checked out a book

Select Name from Patrons **natural left join** CheckedOut **where** CheckedOut.Serial is null

Or

Select Name from Patrons left join CheckedOut **on** Patrons.CardNum=CheckedOut.CardNum **where** CheckedOut.Serial is null

CardNum	Serial
1	1001
1	1004
4	1005
4	1006

Name	CardNum
Joe	1
Ann	2
Ben	3
Dan	4

Nested Query as Condition

- Filter by nested query

```
SELECT x FROM y WHERE x IN (SELECT …);
```

Condition

Nested Query as Condition

- There are several of these operators
- `x` is a value, `A` is a multi-set (e.g. from `SELECT`)

```
x IN A

EXISTS A

x OP y ANY A

x OP y ALL A
```

Nested Query as Condition

- There are several of these operators
- `x` is a value, `A` is a multi-set (e.g. from `SELECT`)

`x IN A:` True if x is in A.

`EXISTS A :` True if A is not empty.

`x OP y ANY A :` True if there exists a y in A such that x op y is True.

`x OP y ALL A :` True if for ALL y in A, x op y is True.

Example

Find all students **not** enrolled in Databases

Students

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B
3	3500	C
3	3810	B
4	3500	C

Courses

cID	Name
3500	SW Practice
3810	Architecture
5530	Databases

Example

- Find all students not enrolled in ‘Databases’

```
select s.Name from Students s
where s.sID not in
(select e.sID from Enrolled e
natural join Courses c
where c.Name='Databases’ );
```

Example

Find all students younger than *everyone* taking Databases

Students

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

Enrolled

sID	cID	Grd
1	3500	A
1	3810	A-
1	5530	A
2	3810	A
2	5530	B
3	3500	C
3	3810	B
4	3500	C

Courses

cID	Name
3500	SW Practice
3810	Architecture
5530	Databases

Example

- Find all students younger than everyone in ‘Databases’

```
select s.sName from Students s
where s.DOB > all
(select DOB from
Students natural join Enrolled
natural join Courses c
where c.Name='Databases');
```

Example

- Find all students younger than everyone in ‘Databases’

```
select s.sName from Students s
where s.DOB > all

    (select s2.DOB from Students s2
   natural join Enrolled natural join
   Courses c where c.cName='Databases’);
```

Example

- Find all students younger than everyone in ‘Databases’

```
select s.sName from Students s
where s.DOB > all

    (select s2.DOB from Students s2
   natural join Enrolled natural join
   Courses c where c.cName='Databases’);
```

- Think of nested queries as nested for-loops

Nested Queries

- Think of nested queries as nested for-loops

```
select s.sName
from Students s
where s.DOB > all
```

```
(select s2.DOB
 from Students s2
 ...);
```

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s.sName
from Students s
where s.DOB > all
```

```
(select s2.DOB
 from Students s2
 ...);
```

s

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s.sName
from Students s
where s.DOB > all
```

```
(select s2.DOB
 from Students s2
 ...);
```

s

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s.sName
from Students s
where s.DOB > all
```

```
(select s2.DOB
 from Students s2
 ...);
```

s

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

- Think of nested queries as nested for-loops

```
select s.sName
from Students s
where s.DOB > all
```

```
(select s2.DOB
 from Students s2
 ...);
```

s

sID	Name	DOB
1	Hermione	1980
2	Harry	1979
3	Ron	1980
4	Malfoy	1982

s2

DOB
1980
1979

Nested Queries

```
select s.sName                    (select s2.DOB
from Students s                    from Students s2
where s.DOB > all                  ...);
```

```
foreach Student s in Students {
  foreach Student s2 in ... {
    if(s.DOB <= s2.DOB)
      don't select s
  }
}
```

EXISTS

- Filter by complex nested query

```
select x from y where
EXISTS
(select …);
```

- If any rows exist in nested query, `x` is selected

EXISTS

- Filter by complex nested query

```
select x from y where
NOT EXISTS
(select …);
```

- If nested query empty (returns nothing), `x` is selected

Division

- Find students taking all classes

Students s

sID	Name
1	Hermione
2	Harry

Courses c

cID	Name
3500	SW Practice
3810	Architecture

Enrolled e

sID	cID
1	3500
1	3810
2	3810

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If the student s is enrolled in the course c, this is non-empty.

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is non-empty, then this is false

If there is an enrollment for {s, c}
Then this select is non-empty

We will not select c if the student is enrolled in it.

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is non-empty, then this is false

If false for all {s, c}, then this select is empty

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

which makes this true

If inner select is non-empty, then this is false

If false for all {s, c}, then this select is empty

If there is an enrollment for {s, c}
Then this select is non-empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

Middle query is basically saying "what are all the courses the student is NOT enrolled in". If this is empty, the student is enrolled in every course.

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

What happens if the student is not enrolled in every course?

If there is **not** an enrollment for {s, c}
Then this select is empty for that {s,c}

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is empty, then this is true

If there is **not** an enrollment for {s, c}
Then this select is empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

If inner select is empty, then this is true

If ever true, then this select is non-empty

If there is **not** an enrollment for {s, c}
Then this select is empty

Division

- Find students taking all classes:

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

which makes this false

If inner select is empty, then this is true

If ever true, then this select is non-empty

If there is **not** an enrollment for {s, c}
Then this select is empty

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{}

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{}

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{}

{3810}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{}

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermione}

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
foreach Course c
foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
foreach Course c
foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

{Hermoine}

foreach Student s
  foreach Course c
    foreach Enrollment e

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermoine}

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermoine}

{3500}

{}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermoine}

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermoine}

(don't add Harry)

{3500}

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810

Division - Example

```
select s.sName
from Students s
where not exists(select c.cID
                 from Courses c
                 where not exists(select e.cID
                                  from Enrolled e
                                  where e.cID = c.cID
                                  and e.sID = s.sID));
```

foreach Student s
foreach Course c
foreach Enrollment e

{Hermoine}

Only student taking all classes

s

sID	Name
1	Hermione
2	Harry

c

cID	Name
3500	SW Practice
3810	Architecture

e

sID	cID
1	3500
1	3810
2	3810