Lecture 4 SQL-Overview

CS211 - Introduction to Database

- Proposed by Boyce and Chamber in 1974
- Implemented by IBM@San Jose for System R,
  - SEQUEL (Structured English QUEry Language)
  - SQL (Structured Query Language)

SQL

- SQL is not a programing language, but rather is a data sub-language
- DDL – Data definition Language
  - Create, Alter, Drop
  - Schema: database, table, view, integrity, index
- DML – Data manipulation language
  - Insert, Delete, Update, Select
  - Data: append from input or other tables, query, update
- DCL – Data control language
  - Creating user accounts - Grant, revoke
  - Security control

Sample Database

- Student(sID char(8), sName char(10), gender char(1), age int, dID char(2), grade char(10))
- Dept(dID char(2), dName char(10), dean char(10))
- Course(cID char(3), cName char(12), hours int, credit int, iID char(3))
- Instructor(iID int, iName char(10), dID char(2), workload float)
- RC(sID char(8), cID char(3), score float)

SQL for Data Definition

- The statements include
  - CREATE
    - To create database objects: database, table, …
  - ALTER
    - To modify the structure and/or characteristics of existing database objects: add new attributes …
  - DROP
    - To delete database objects: table, view, …

SQL for Data Definition - CREATE

- To create a new database, the SQL statement

CREATE DATABASE *database_name*

e.g. CREATE DATABASE school

SQL for Data Definition - CREATE

- To create tables, the SQL statement

```
CREATE TABLE student (
    sID char(8) NOT NULL,
    sName char(10) NOT NULL,
    gender char(1),
    age int,
    dID char(2),
    grade char(10)
);
```

```
CREATE TABLE student (
    sID char(8) PRIMARY KEY,
    sName char(10) NOT NULL,
    gender char(1),
    age int,
    dID char(2),
    grade char(10)
);
```

SQL for Data Definition - CREATE

- To create tables, the SQL statement
- Composite PRIMARY KEY

```
CREATE TABLE rc (
   sID char(4),
   cID char(3),
   score float,
   PRIMARY KEY(sID,cID)
);
```

Data type

- char(n): string with fixed length
- varchar(n): string with variable length
- int/integer
- numeric(p,q)   [p-q digits].[q digits]
- real/float(n)   rounded up to *n* decimal places
- date 15-05-2018
- time 23:55:000

SQL for Data Definition - DROP

DROP DATABASE *database_name*

e.g. DROP DATABASE school

DROP TABLE *table_name*

e.g. DROP TABLE student

DROP TABLE IF EXISTS student;

SQL for Data Manipulation

- The statements modifying data include
  - INSERT INTO
    - To append a new tuple to a table
  - UPDATE
    - To update the rows in a table which match the specified criteria
  - DELETE
    - To delete the rows in a table which match the specified criteria
- The statements querying data
  - SELECT

Append tuples

```
Student(sID char(4) not null,
            sName char(10),
            gender char(1),
            age int dID char(2),
            grade char(10)  )
```

```
INSERT INTO Student VALUES('4001', 'Amy', 'F', 20, '03', 'Sophomore');
```

```
INSERT INTO Student(sID, sName, gender, dID)
VALUES('4002', 'Alice', 'F', '04');
```

Append tuples - batch

```
St (sID char(4) not null,
    sName char(10) )
```

```
INSERT INTO St(sID, sName)
   SELECT sID, sName FROM student WHERE sName like '%y';
```

Update tuples

```
Student(sID char(4) not null,
              sName char(10),
              gender char(1),
              age int, dID char(2),
              grade char(10)  )
```

```
UPDATE Student SET age=19 WHERE sID='4001';
```

Update tuples

```
Instructor(iID int,
                iName char(10),
                dID char(2),
                workload float)
```

Increase 10% workload for all instructors in CS

```
UPDATE Instructor SET workload=workload*1.1
WHERE dID=‘03’;
```

Delete tuples

```
Student(sID char(4) not null,
        sName char(10),
        gender char(1),
        age int, dID char(2),
        grade char(10)  )
```

```
DELETE from Student WHERE sID='4001';
```

Simple query

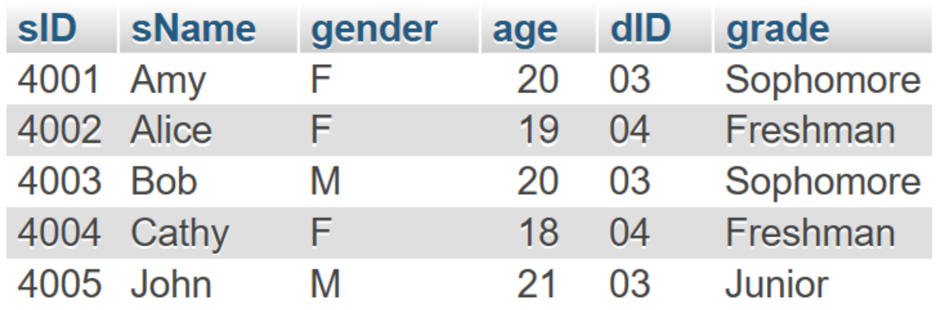
SELECT $col1, \ldots, coln$

FROM table

[WHERE query_condition];

$$\boxed{\pi_{col1,\ldots,coln}(\sigma_{query\_condition}(table))}$$

Query student



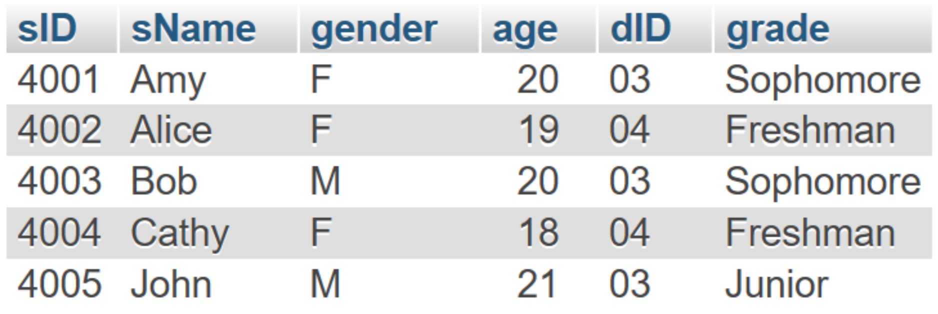
sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Return all student’s information

- Return name and age for all students

- Return age and name for students who are under 19

Query student



sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

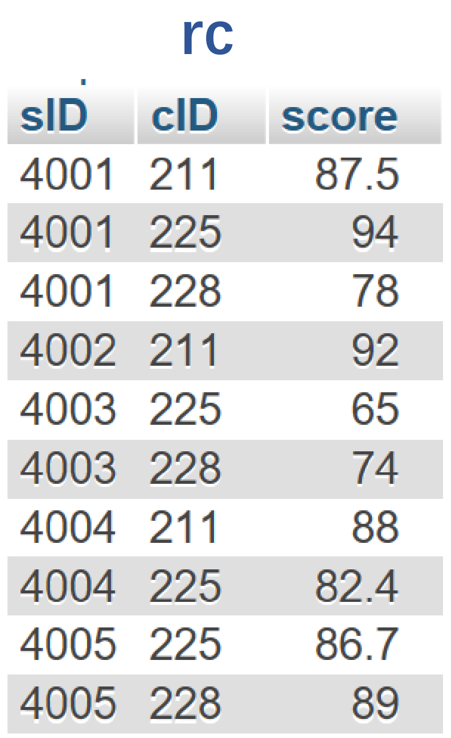
- Return all student’s information
  - SELECT * FROM students;
- Return name and age for all students
  - SELECT sName, age FROM students;
- Return age and name for students who are under 19
  - SELECT sName, age FROM students WHERE age < 19;

Multiple matching criteria

Use AND, OR, NOT to connect criteria

- Return the id of students who takes course either '211' or ‘225'

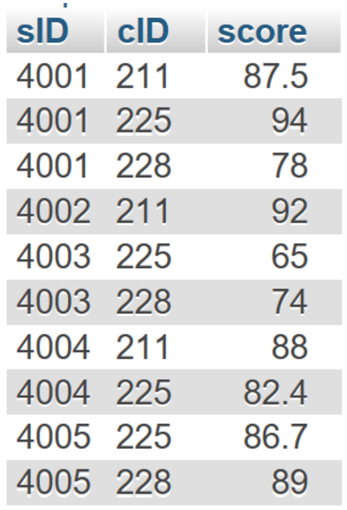
- Return the id of students who takes both '211' and '225'



rc

sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

Multiple matching criteria



sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

rc

- Return the id of students who takes course either '211' or ‘225'
  - SELECT [distinct] rc.sID FROM rc WHERE cID=‘211’ OR cID=‘225’;
- Return the id of students who takes both '211' and '225'
  - SELECT distinct rc.sID FROM rc WHERE cID=‘211’ AND cID=‘225’; Incorrect
  - Solution 1: select distinct sID from rc where cID=‘211’ intersect
    select distinct sID from rc where cID=‘225’;
  - Solution 2: select rc.sID from rc, rc as rc1 where rc.sid=rc1.sid and rc.cid='211' and rc1.cid='225';
  - Solution 3: use Keyword IN

- distinct
  - Return the id of students who has a score > 80

- order by col[asc|desc]
  - Return all students' id and name in ascending order by student ID

  - Return scores of course '211' and list them from high to low

- DISTINCT
  - Return the id of students who has a score > 80
    - SELECT DISTINCT sID FROM rc WHERE score > 80;

- ORDER BY col[ASC|DESC]
  - Return all students’ id and name in ascending order by student ID
    - SELECT sID, sName FROM student ORDER BY sID;

  - Return scores of course '211' and list them from high to low
    - SELECT sID, score FROM rc WHERE cID=‘211’ ORDER BY score DESC;

SQL wildcard

sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- like / not like
  - %: zero, one, or multiple characters
  - _: a character

```
SELECT sID, sName FROM Student WHERE sName LIKE 'A%' ;
```

```
SELECT sID, sName FROM Student WHERE sName LIKE 'A_ _' ;
```

```
SELECT sID, sName FROM Student WHERE sName NOT LIKE 'A%' ;
```

Query multiple tables

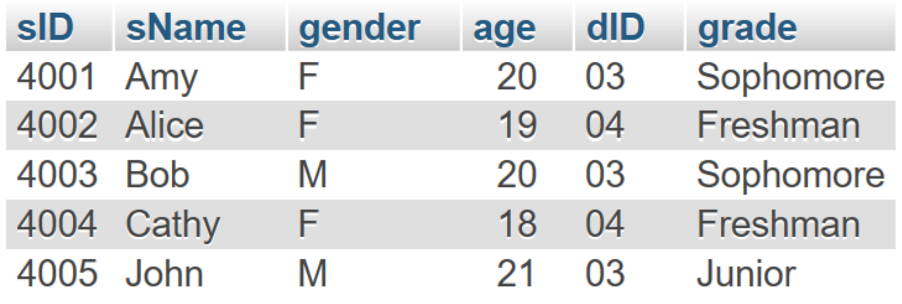
SELECT $col1, \dots, coln$

FROM table1, table2, ⋯

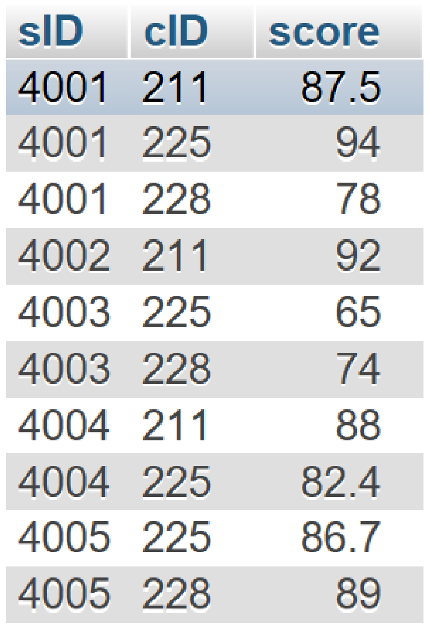
[WHERE query_condition];

$$\boxed{\pi_{col1,\dots,coln}(\sigma_{condition}(table1 \times table2 \dots))}$$

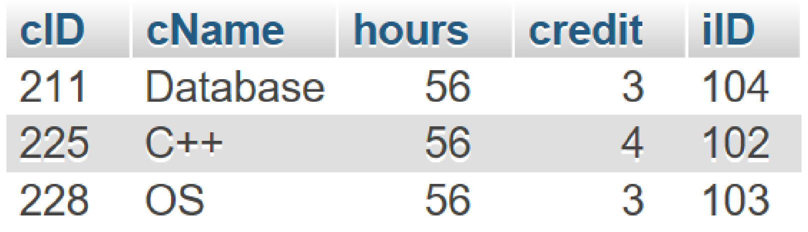
- Return the scores of course 'database' for all students, list students' name, score, and sort the result in descending order by score



sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior



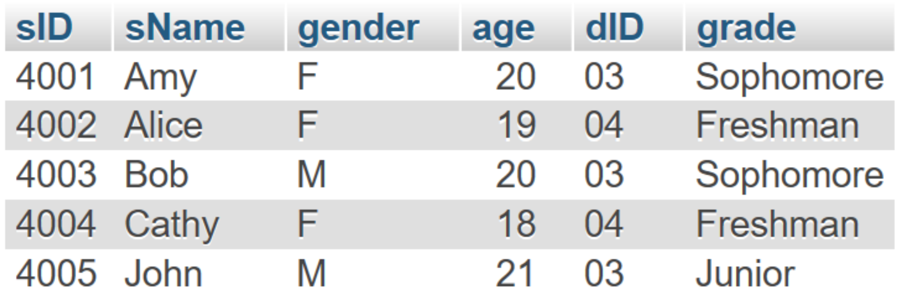
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



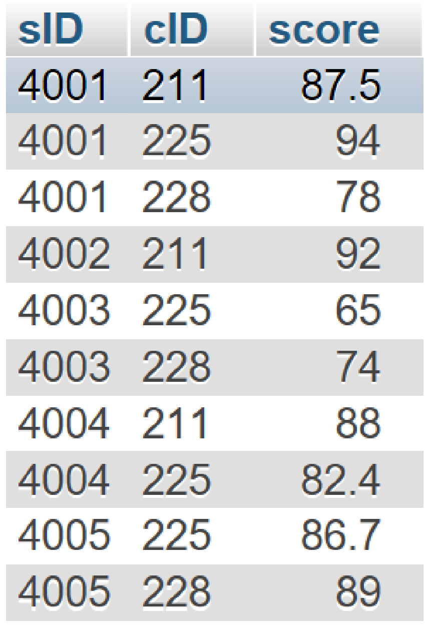
cID	cName	hours	credit	iID
211	Database	56	3	104
225	C++	56	4	102
228	OS	56	3	103

- Return the scores of course 'database' for all students, list students’ name, score, and sort the result in descending order by score

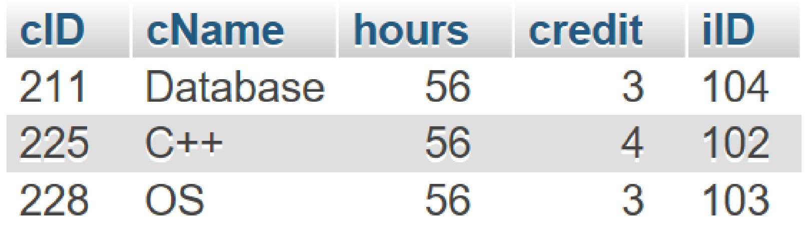
```
SELECT sName, score FROM student, rc, course
WHERE cName=‘database’ AND course.cid=rc.cid AND
rc.sid=student.sid
ORDER BY score DESC;
```



sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior



sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



cID	cName	hours	credit	iID
211	Database	56	3	104
225	C++	56	4	102
228	OS	56	3	103

Alias

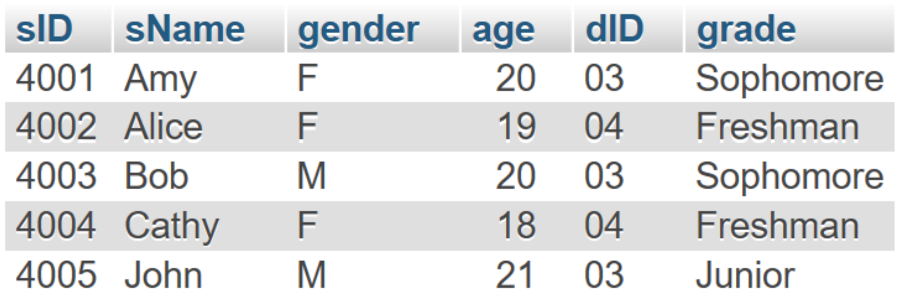
$\boxed{\rho_{R1}(R)}$

SELECT col as alias [[, col as alias]⋯]
FROM table1 alias1, table2 as alias2, ⋯
[WHERE query_condition];

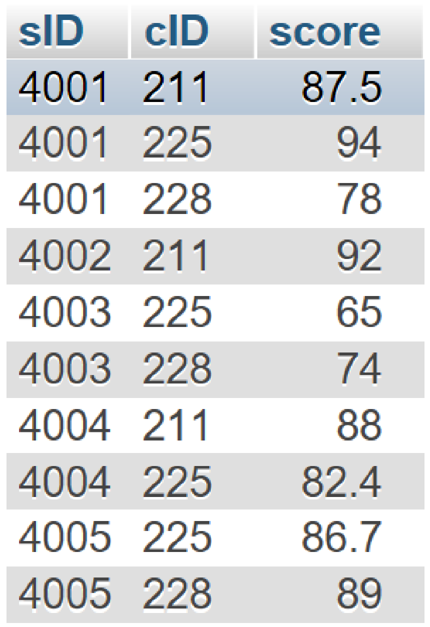
When:

- $*$ TableA join TableA
- $*$ long table name

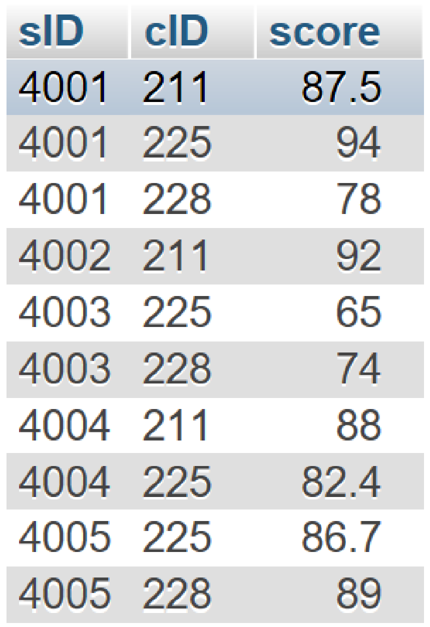
- Find students(name) who take both courses '211' and '225'



sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior



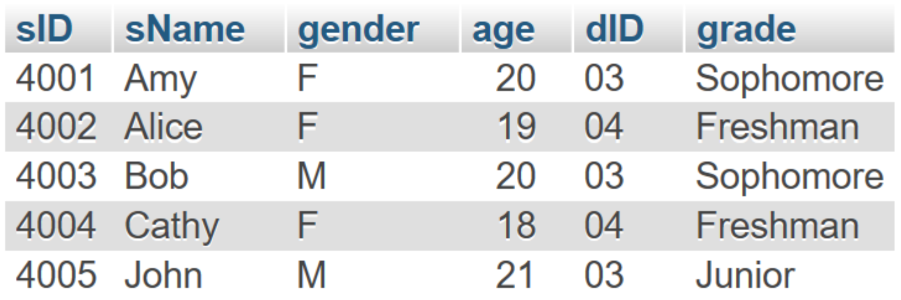
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



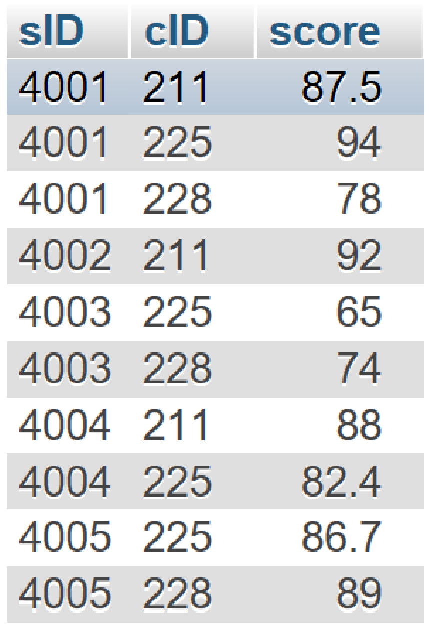
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

- Find students(name) who take both courses '211' and '225'

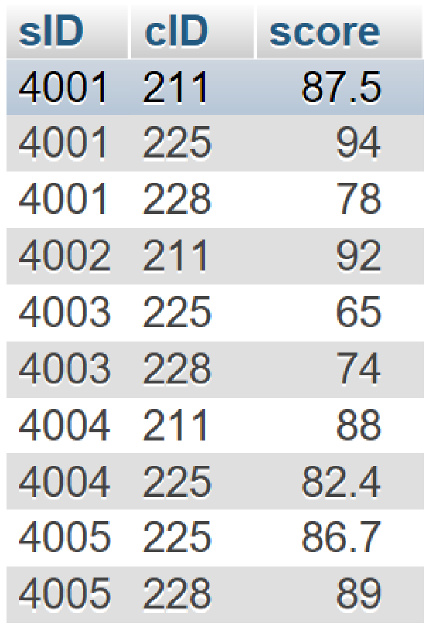
```
SELECT student.sName FROM rc r1, rc r2, student
WHERE r1.sID=r2.sID AND r1.cID=‘211’ AND r2.cID=‘225’ AND
              student.sID = r1.sID;
```



sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

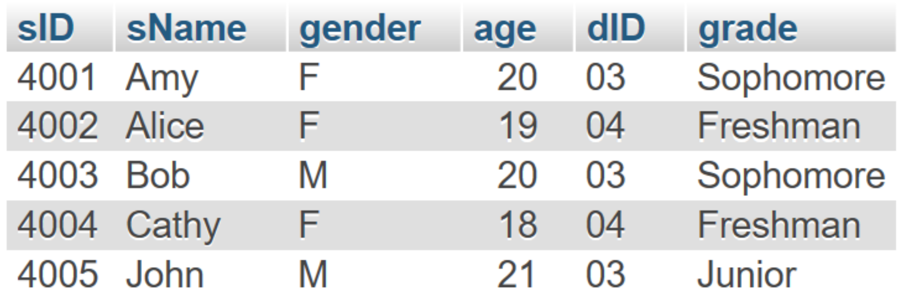


sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

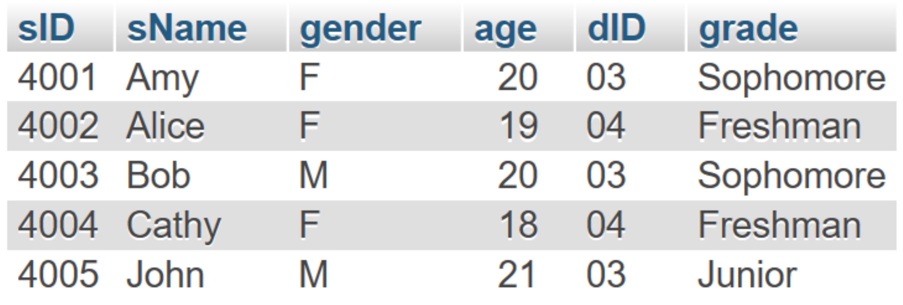


sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

- Find all pairs of students (A,B) where A’s age is not same as B’s.



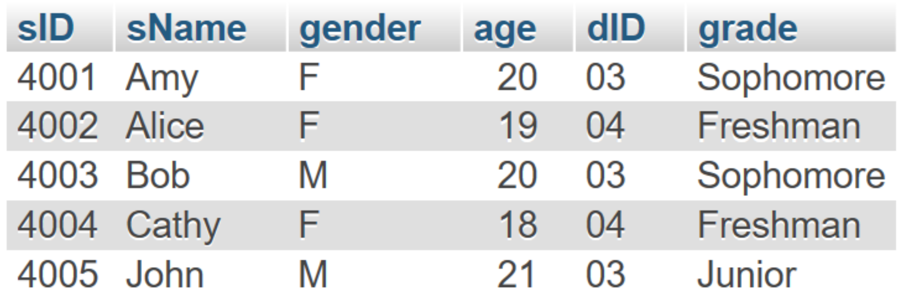
sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior



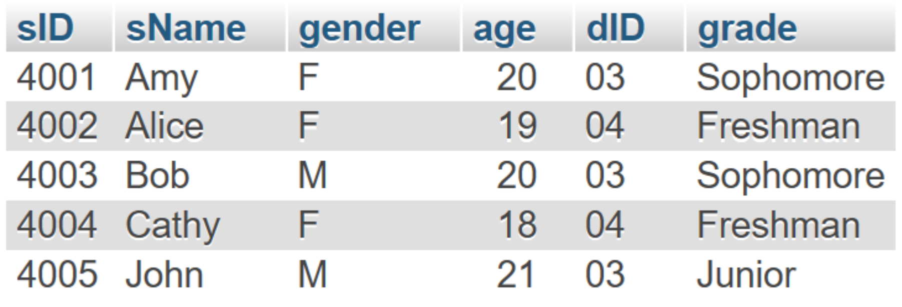
sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Find all pairs of students (A,B) where A’s age is not same as B’s.

```
SELECT s1.sName, s2.sName FROM student s1, student s2
WHERE s1.sID<s2.sID AND s1.age <> s2.age;
```

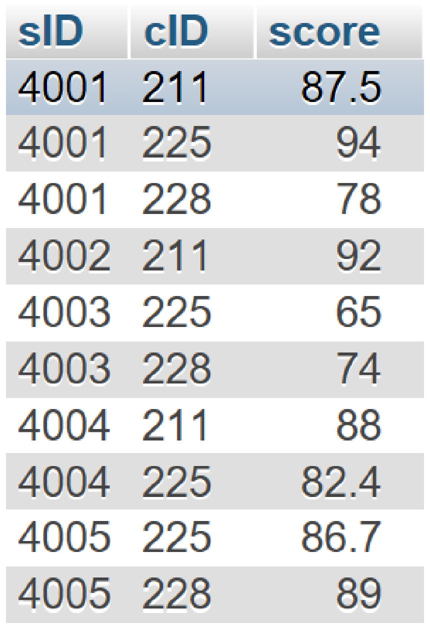


sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

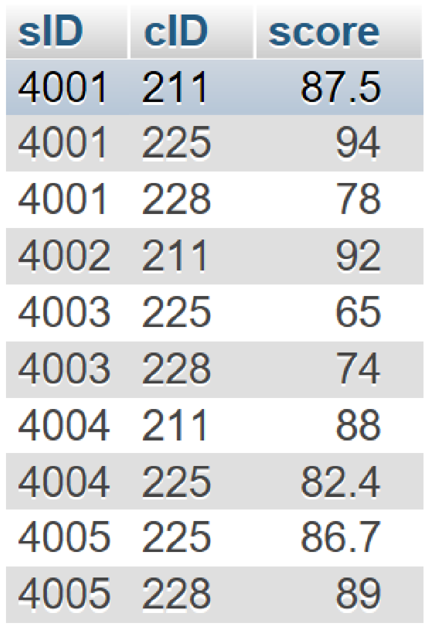


sID	sName	gender	age	dID	grade
4001	Amy	F	20	03	Sophomore
4002	Alice	F	19	04	Freshman
4003	Bob	M	20	03	Sophomore
4004	Cathy	F	18	04	Freshman
4005	John	M	21	03	Junior

- Find students(sID) whose score of '211' is higher than that of '225'



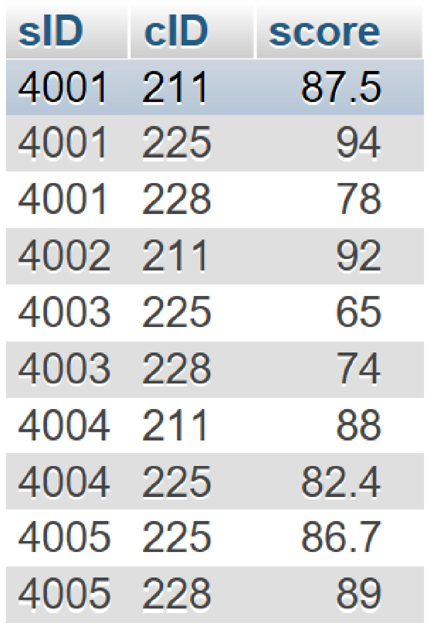
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



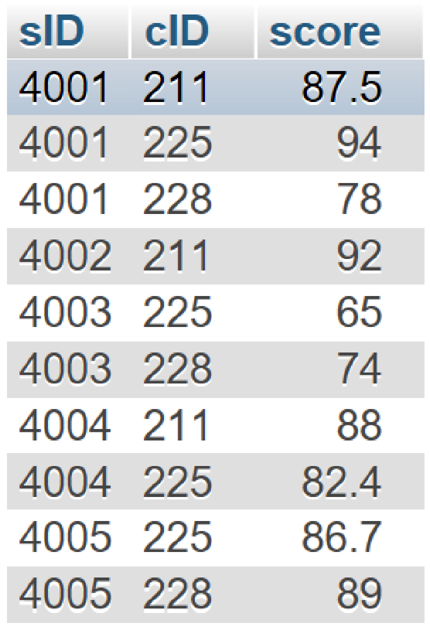
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

- Find students(sID) whose score of '211' is higher than that of '225'

```
SELECT r1.sID FROM rc r1, rc r2
WHERE r1.sID=r2.sID AND r1.cID='211' AND
            r2.cID = '225' AND r1.score > r2.score;
```



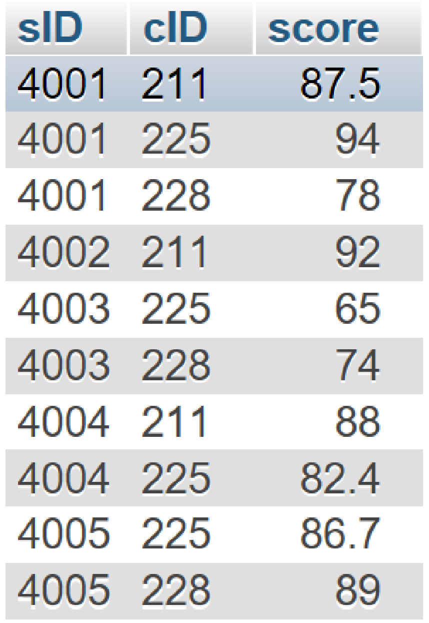
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



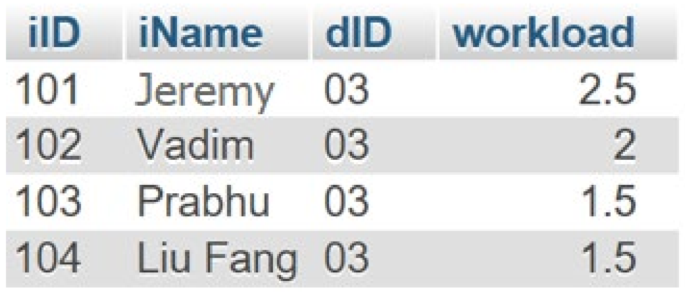
sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89

- Find students(sID) who never took any course lectured by ‘Prabhu'

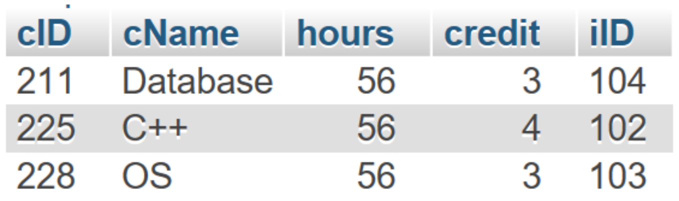
```
SELECT rc.sID FROM rc, course c, instructor i
WHERE rc.cID=c.cID AND c.iID=i.iID AND
            i.iName!=‘Prabhu’;       ×
```



sID	cID	score
4001	211	87.5
4001	225	94
4001	228	78
4002	211	92
4003	225	65
4003	228	74
4004	211	88
4004	225	82.4
4005	225	86.7
4005	228	89



iID	iName	dID	workload
101	Jeremy	03	2.5
102	Vadim	03	2
103	Prabhu	03	1.5
104	Liu Fang	03	1.5



cID	cName	hours	credit	iID
211	Database	56	3	104
225	C++	56	4	102
228	OS	56	3	103