/* Students enroll to Modules database*/

```
CREATE TABLE Student(
sID INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
sName VARCHAR(50) NOT NULL,
GPA TINYINT
);

CREATE TABLE Module(
mCode CHAR(6) NOT NULL PRIMARY KEY,
mCredits TINYINT NOT NULL DEFAULT 10,
mTitle VARCHAR(100) NOT NULL
);

CREATE TABLE Grade(
sID INT,
mCode CHAR(6),
gMark TINYINT,
CONSTRAINT pk1 PRIMARY KEY(sID, mCode),
CONSTRAINT fk1 FOREIGN KEY (sID) REFERENCES Student (sID) ON DELETE CASCADE,
CONSTRAINT fk2 FOREIGN KEY (mCode) REFERENCES Module (mCode) ON DELETE CASCADE
);


INSERT INTO Student
(sID, sName, GPA)
VALUES
(1, 'John', 18.5),
(2, 'Mary', 19),
(3, 'James', 18),
(4, 'Amy', 17),
(5, 'John', 18.5),
(6, 'Amy', 18);

INSERT INTO Module
VALUES
('G51DBI', 10, 'Databases and Interfaces'),
('G51PRG', 20, 'Programming'),
('G51IAI', 10, 'Artificial Intelligence'),
('G52ADS', 10, 'Algorithms');


INSERT INTO Grade
(sID, mCode, gMark)
VALUES
(1, 'G51DBI', 70),
(1, 'G51PRG', 60),
```

(1, 'G51IAI', 60),
(2, 'G51DBI', 80),
(2, 'G51PRG', 50),
(2, 'G51IAI', 60),
(3, 'G51DBI', 50),
(3, 'G51PRG', 50),
(3, 'G51IAI', 60),
(4, 'G51DBI', 75),
(4, 'G51PRG', 65),
(4, 'G51IAI', 55),
(5, 'G51DBI', 70),
(5, 'G51PRG', 50),
(5, 'G51IAI', 50),
(6, 'G51DBI', 70),
(6, 'G51PRG', 100),
(6, 'G51IAI', 55);

%%%%%%%%%%%%%%%%%%%
Quick revision

select sname, gpa
from Student

- add where gpa>17;
- illustrate distinct, *

%%%%%%%%%%%%%%%%%%%
Cartesian product

select *
from student, grade;
>> result has 108 rows

>> Add constraint and project:

select sname, gmark
from student, grade
where student.sid = grade.sid;

/* Find "Names and GPAs of students with mark in Databases > 60 " */

select sname, gpa
from student, grade
where student.sid = grade.sid and mcode = 'G51DBI' and gmark > 60;

>> add sid (won't work), add student.sid to fix it

/* Find " IDs, names and GPAs of students, and their marks and credits for the modules they have enrolled" */

```
select student.sid, sname, gpa, grade.mcode, gmark, mcredits
from student, grade, module
where student.sid = grade.sid and grade.mcode = module.mcode;
```

>> nice example illustrating that the new table is grade table plus one column from student and one from module!

- order by

```
select student.sid, sname, gpa, grade.mcode, gmark, mcredits
from student, grade, module
where student.sid = grade.sid and grade.mcode = module.mcode
order by student.sid desc;
```

>> add: gmark asc;

Table variables

- start from:

```
select sname, gpa
from student, grade
where student.sid = grade.sid and mcode = 'G51DBI' and gmark > 60;
```

>> change to

```
select s.sid, sname, gpa
from student as s, grade as g
where s.sid = g.sid and mcode = 'G51DBI' and gmark > 60;
```

- we can rename result too

```
select s.sid, sname, 5*gpa
from student as s, grade as g
where s.sid = g.sid and mcode = 'G51DBI' and gmark > 60;
```

>> change to: 5*gpa as normalized_gpa

- **/*"Find all pairs of students with the same GPA" */**

```
select s1.sname, s2.sname, s1.gpa, s2.gpa
from student s1, student s2
where s1.gpa = s2.gpa and s1.sid > s2.sid;
```

>> illustrate that we have 2 different Johns with the same name

Subqueries:

- Use the result of a query as input to a new query (reminiscent of ??)

/* find the name and gpa of the student with the highest sid who has enrolled to some module */

```
select max(sid)
from Grade

select sname, gpa
from student
where sid = (select max(sid)from grade)
```

>> different way to combine the data from two table

- Vs cross product:

/* "Find the sIDs and names of students who have at least one module with mark > 60" */

```
select s.sid, sname
from student s, grade g
where s.sid = g.sid and gmark> 60
```

>> remove duplicates with distinct

>> now remove s.sid

```
select sname
from student s, grade g
where s.sid = g.sid and gmark> 60
```

>> removing duplicates with distinct removes more than we wanted

>> alternative with subquery:

>>find first all sid for which gmark>60

```
select distinct sid from grade where gmark> 60
```

>> then form subquery, first try with =

```
select sname
from student s
where s.sid = (select distinct sid from grade where gmark> 60)
```

>> does not work then replace = with IN

```
select sname
from student s
where s.sid in (select distinct sid from grade where gmark> 60)
```

/* "Find the sIDs and names of students who have at least one module with mark > 70 but <80" *?

```
select distinct sname, s.sid
from student s, grade g
where s.sid = g.sid and gmark > 70 and gmark<80
```

>> using in and not in (this is the difference operator)

```
select sname, sid
from student
where sid in (select sid from grade where gmark > 70)
and sid not in (select sid from grade where gmark >= 80)
```

EXISTS

/* "Find the names and IDs of all students with the same name" */

>> with self join

```
select s1.sid, s1.sname, s2.sid, s2.sname
from student s1, student s2
where s1.sname = s2.sname and s1.sid>s2.sid
```

```
select sid, sname
from student s1
where exists (select * from student s2 where s1.sid > s2.sid and s1.sname = s2.sname)
```

/* "Find highest mark from Grade" */

>> MAX without max()

```
select sid, gmark
from grade g1
where exists (select * from grade g2 where g1.gmark>g2.gmark)
```

>> The above does not work. It returns all marks which are higher than some other mark. So only the minimum mark is not included!

```
select sid, gmark
from grade g1
where exists (select * from grade g2 where g1.gmark<g2.gmark)
```

>> only the maximum is not included

```
select sid, gmark
from grade g1
where not exists (select * from grade g2 where g1.gmark<g2.gmark)
```

ALL

>> allows to use equalities/inequalities with sets

```
select sid, gmark
from grade g1
where g1.gmark > (select g2.gmark from grade g2)
```

>> the above won't work, use all to fix it

```
select sid, gmark
from grade g1
where g1.gmark > all(select g2.gmark from grade g2)
```

>> using >= to get the highest

```
select sid, gmark
from grade g1
where g1.gmark >= all(select g2.gmark from grade g2)
```

ANY

```
select sid, gmark
from grade g1
where g1.gmark >= any(select g2.gmark from grade g2)
```

>> will return all grades because of >=

```
select sid, gmark
from grade g1
where g1.gmark >any(select g2.gmark from grade g2)
```

>> will return all grades but the minimum