/* 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);
%%%%%%%%%%%%%%%%

Join Family of Operators:
>> Cross Product

/* "Find sIDs and names of all students who enrolled to some module" */
select distinct s.sid, sname
from student s, grade g
where s.sid = g.sid

>> Re-write using inner-join (equiv to theta join)

select distinct s.sid, sname
from student s inner join grade g
on s.sid = g.sid

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

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

>> Replace WHERE with AND (many diff ways to express the same query)

select sname, gpa
from student s join grade g
on s.sid = g.sid and gmark>60 and mcode = 'G51DBI'

>> Introduce Natural Join

select distinct s.sid, sname
from student s natural join grade g

>> Natural Join eliminates columns with the same attribute name, so we can remove Student from Student.sID

```
select distinct sid, sname
from student s natural join grade g
```

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

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

>> Introducing JOIN USING

```
select sid, sname, gpa
from student s  join grade g using (sid)
where gmark>60 and mcode = 'G51DBI'
```

>> more than 2 tables

```
select *
from student s join grade g using (sid) join module using (mcode)
```

>> Introducing Left Outer Join >> Insert new data

```
INSERT INTO Student VALUES
(7, 'George', 20)
```

```
select * from student join grade using (sid);
```

vs

```
select * from student left outer join grade using (sid);
```

>> right outer join

```
select * from student right outer join grade using (sid);
```

vs

```
select * from grade right outer join student using (sid);
```

>> Rewrite Left OUTER JOIN with other operators

```
(select s.sid, sname, gpa, mcode, gmark
from student s, grade g
where s.sid= g.sid)
union
(select s.sid, sname, gpa, null, null
from student s
where s.sid not in (select sid from grade))
```

Aggregation: perform computations over sets of values in multiple rows of relations

/* "Find min, max, avg, count of GPA of all students with mark in G51DBS >=60" */

```
select gpa
from student s
where sid in (select sid from grade where gmark>60 and mcode = 'G51DBI')
```

>> all of them return a single value
>> be careful with combining aggregation with other columns

```
select sid, gpa
from student s
where sid in (select sid from grade where gmark>60 and mcode = 'G51DBI')
```

>> if min(gpa) is used, result is not the expected one

```
select sid, min(gpa)
from student s
where sid in (select sid from grade where gmark>60 and mcode = 'G51DBI')
```

>> correct way of doing this:

```
select sid
from student where gpa = (select min(gpa) from student s where sid in (select sid from grade
where gmark>60 and mcode = 'G51DBI'))
```

>> aggregation can be combined with distinct

```
select (sid) from grade where gmark>60
```

>> the following will count duplicates

```
select count(distinct sid) from grade where gmark>60
```

Introduce GROUP BY clause (used in conjunction with aggregation)
/ * "Find the number of Students who have selected each module" */

```
select count(sid)
from grade
group by mcode
```

/ * "Find the max grade for each module" */

```
select max(gmark)
from grade
group by mcode
```

>> add mcode, nice!

```
select max(gmark), mcode
from grade
group by mcode
```

```
>> select *
from grade
group by mcode
```

>> notice that only mcode makes sense from the result. To get something meaningful from the other columns we need to apply aggregation. Otherwise the first row is returned.

/* Find how many students have taken a specific mark for each module /*

```
select gmark ,mcode, count(gmark)
from grade
group by gmark, mcode
```

/* Find how many students have taken a specific mark for each module. Only marks greater than 50 should be considered /*

```
>> introduce having
select gmark ,mcode, count(gmark)
from grade
group by gmark, mcode
having gmark > 50
```

>> the constraint in having must include a column specified in group by. The following won't work

```
select max(gmark)
from grade
group by mcode
having gmark > 60
```

>> however the following will work:

/* Find the average for each module for which the average is > 60 /*

```
select mcode, avg(gmark)
from grade
group by mcode
having avg(gmark) > 60
```

/* Find how many students have taken above the average mark for each module /*

```
select mcode, count(diff)
from (select g.mcode, (g.gmark - temp.avg_mark) as diff
from grade g, (select mcode, avg(gmark) as avg_mark from grade group by mcode) as temp
where g.mcode = temp.mcode) as temp2
where temp2.diff>=0
group by mcode
```