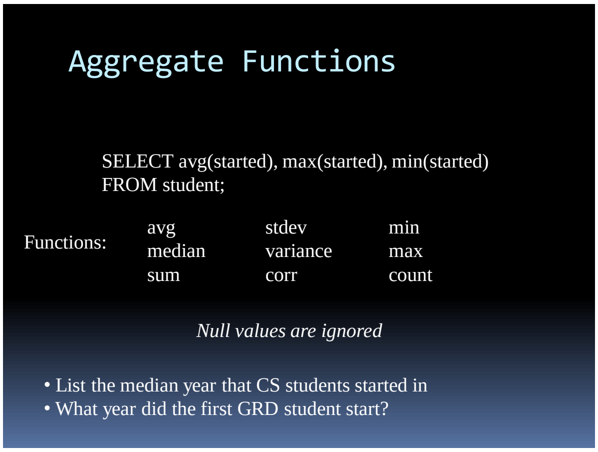
ADVANCED SQL

Aggregation



Aggregate Functions

```
SELECT avg(started), max(started), min(started)
FROM student;
```

Functions:

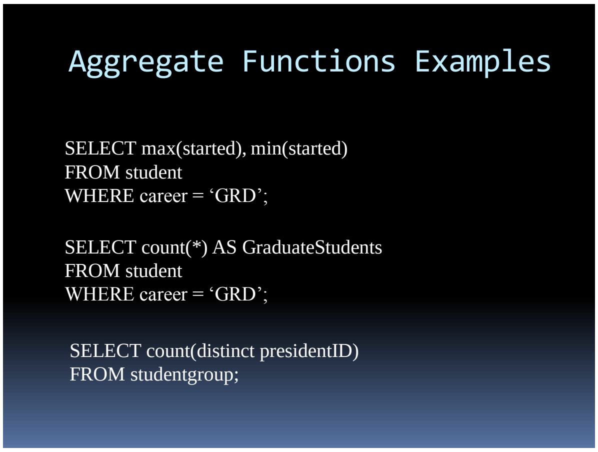
avg
median
sum

stdev
variance
corr

min
max
count

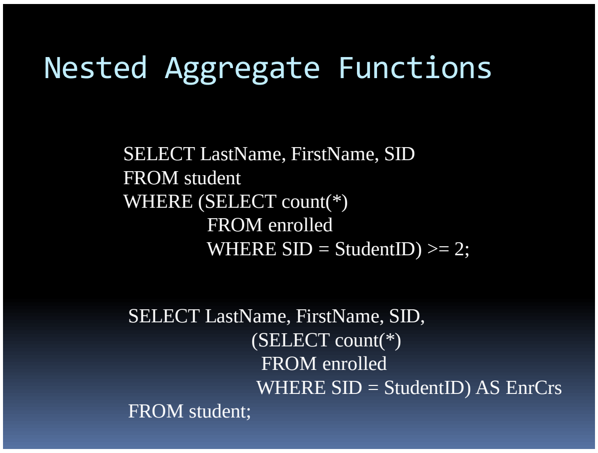
Null values are ignored

- List the median year that CS students started in
- What year did the first GRD student start?



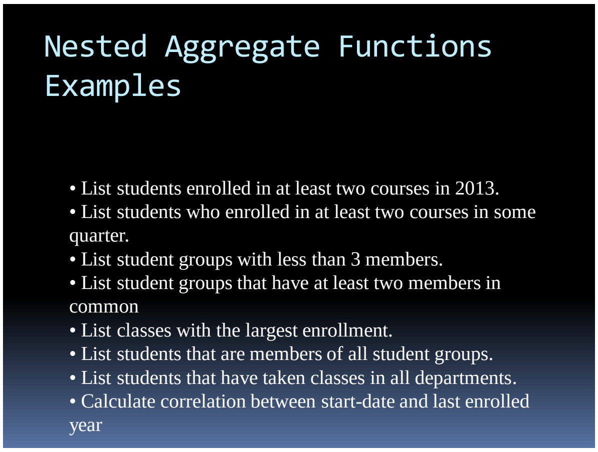
Aggregate Functions Examples

```
SELECT max(started), min(started)
FROM student
WHERE career = ‘GRD’;

SELECT count(*) AS GraduateStudents
FROM student
WHERE career = ‘GRD’;

SELECT count(distinct presidentID)
FROM studentgroup;
```



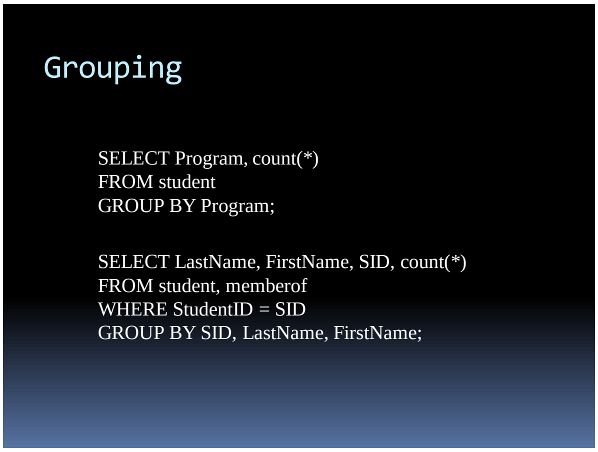
Nested Aggregate Functions

```
SELECT LastName, FirstName, SID
FROM student
WHERE (SELECT count(*)
          FROM enrolled
          WHERE SID = StudentID) >= 2;

SELECT LastName, FirstName, SID,
               (SELECT count(*)
                FROM enrolled
               WHERE SID = StudentID) AS EnrCrs
FROM student;
```



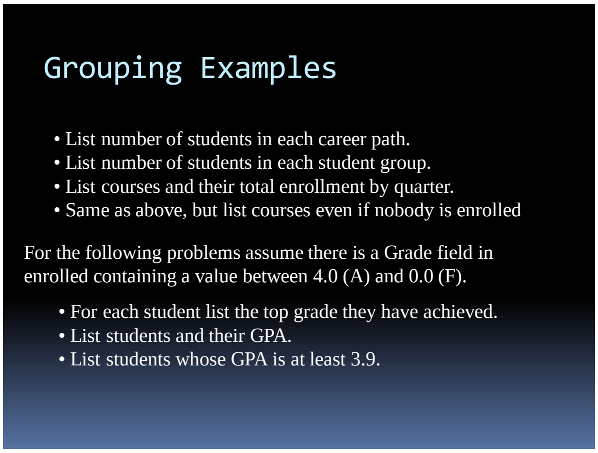
Nested Aggregate Functions Examples

- List students enrolled in at least two courses in 2013.
- List students who enrolled in at least two courses in some quarter.
- List student groups with less than 3 members.
- List student groups that have at least two members in common
- List classes with the largest enrollment.
- List students that are members of all student groups.
- List students that have taken classes in all departments.
- Calculate correlation between start-date and last enrolled year



Grouping

```
SELECT Program, count(*)
FROM student
GROUP BY Program;

SELECT LastName, FirstName, SID, count(*)
FROM student, memberof
WHERE StudentID = SID
GROUP BY SID, LastName, FirstName;
```

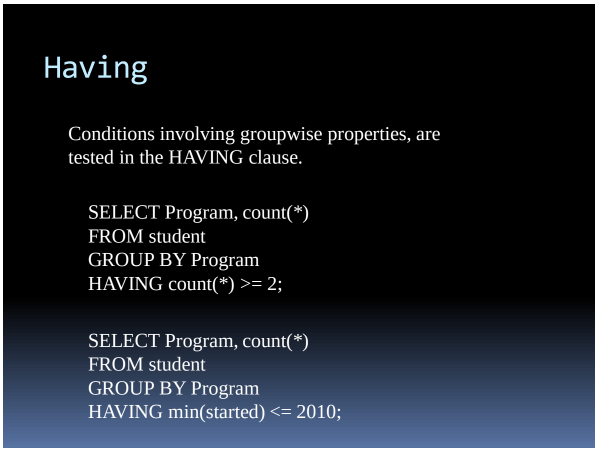


Grouping Examples

- List number of students in each career path.
- List number of students in each student group.
- List courses and their total enrollment by quarter.
- Same as above, but list courses even if nobody is enrolled

For the following problems assume there is a Grade field in enrolled containing a value between 4.0 (A) and 0.0 (F).

- For each student list the top grade they have achieved.
- List students and their GPA.
- List students whose GPA is at least 3.9.



Having

Conditions involving groupwise properties, are tested in the HAVING clause.

```
SELECT Program, count(*)
FROM student
GROUP BY Program
HAVING count(*) >= 2;

SELECT Program, count(*)
FROM student
GROUP BY Program
HAVING min(started) <= 2010;
```

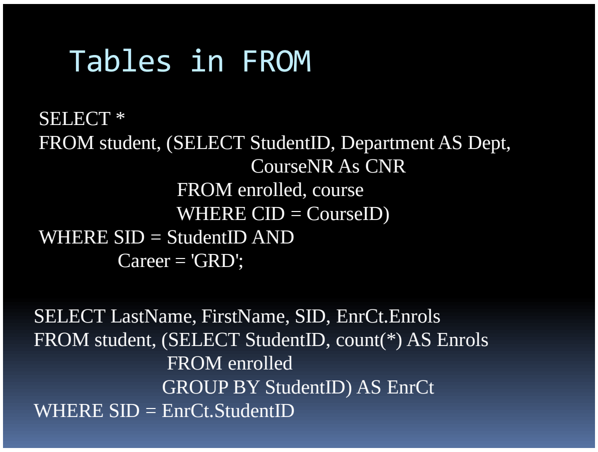
Having Examples

- List students whose GPA is at least 3.9.
- List courses in which at least two students are enrolled.
- List departments in which the average enrollment in courses is below 2
- For each program compute the number of Chicago students in the program but only include programs that have at least three students.

SELECT Syntax

SELECT attributes and functions (define aliases)
FROM list of tables (define aliases)
WHERE condition
GROUP BY grouping attributes
HAVING group condition
ORDER BY attribute list

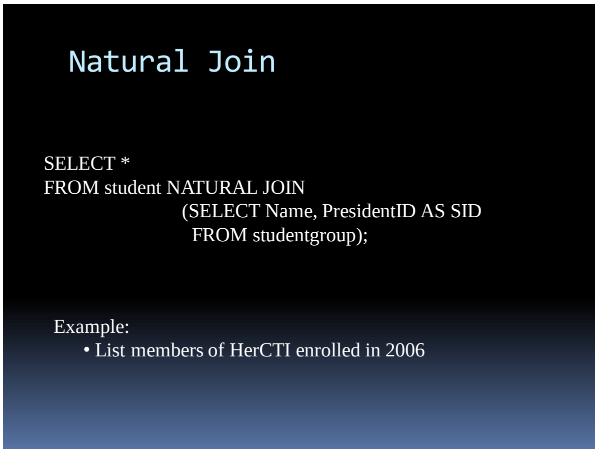
Odds & Ends



Tables in FROM

```
SELECT *
FROM student, (SELECT StudentID, Department AS Dept,
                      CourseNR As CNR
               FROM enrolled, course
               WHERE CID = CourseID)
WHERE SID = StudentID AND
        Career = 'GRD';
```

```
SELECT LastName, FirstName, SID, EnrCt.Enrols
FROM student, (SELECT StudentID, count(*) AS Enrols
               FROM enrolled
               GROUP BY StudentID) AS EnrCt
WHERE SID = EnrCt.StudentID
```

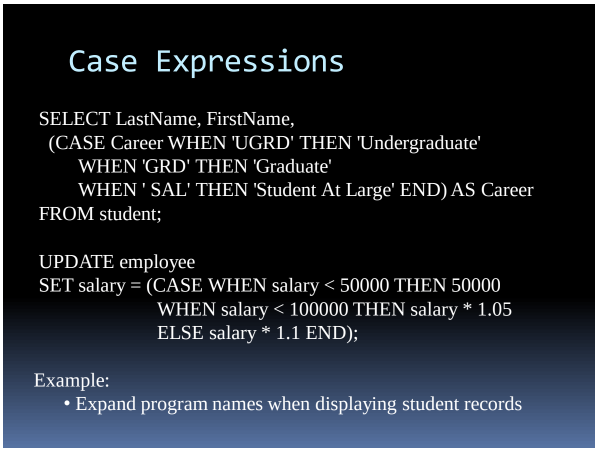


Natural Join

```
SELECT *
FROM student NATURAL JOIN
              (SELECT Name, PresidentID AS SID
               FROM studentgroup);
```

Example:

- List members of HerCTI enrolled in 2006



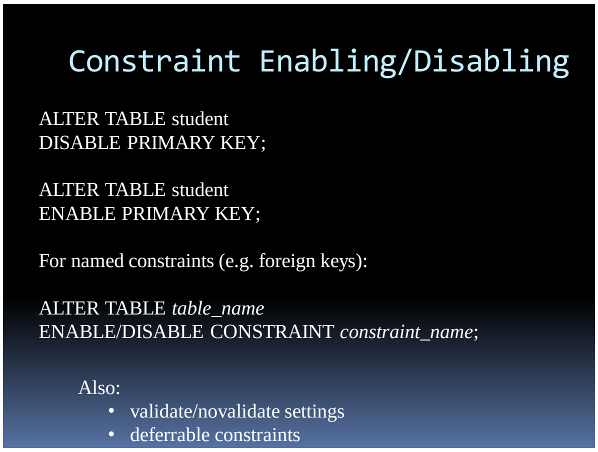
Case Expressions

```
SELECT LastName, FirstName,
 (CASE Career WHEN 'UGRD' THEN 'Undergraduate'
    WHEN 'GRD' THEN 'Graduate'
    WHEN ' SAL' THEN 'Student At Large' END) AS Career
FROM student;
```

```
UPDATE employee
SET salary = (CASE WHEN salary < 50000 THEN 50000
              WHEN salary < 100000 THEN salary * 1.05
              ELSE salary * 1.1 END);
```

Example:

- Expand program names when displaying student records



Constraint Enabling/Disabling

ALTER TABLE student
DISABLE PRIMARY KEY;

ALTER TABLE student
ENABLE PRIMARY KEY;

For named constraints (e.g. foreign keys):

ALTER TABLE *table_name*
ENABLE/DISABLE CONSTRAINT *constraint_name*;

Also:

- validate/novalidate settings
- deferrable constraints