$\gamma_L(R)$ – grouping and aggregation

SQL query returning a #, or a sequence of #:

$\gamma_{\quad}(R)$

all the elements of the subscript are under aggr fns

© North Carolina State University, All Rights Reserved

```
select max(GPA), avg(SAT)
from Student;
```

```
γ max(GPA), avg(SAT) (Student)
```

© North Carolina State University, All Rights Reserved

Total # of applications to the CS major

select count(major)
(1) from Apply
(2) where major = 'CS'

(3) $\gamma_{count(major)}$ ((2) $\sigma_{major='CS'}$ ((Apply) C))

© North Carolina State University, All Rights Reserved

by email

Return the IDs of all the students whose GPA is the top GPA among all the students

```
select ID
from Student
where GPA = max(GPA)
```

© North Carolina State University, All Rights Reserved

Evaluating a $\gamma_L(R)$

[1] Partition R by values of the unaggregated attributes in the subscript L of γ

	A	B	C
(1,2) →	1	2	3
	1	2	7
	1	2	7
(4,5) →	4	5	1
	4	5	3
(4,6) →	4	6	2

© North Carolina State University, All Rights Reserved

[2] Within each individual partition, apply all the aggregation functions in the subscript L of γ_L

[3] For each individual partition, form and output a single tuple, with its values of all the unaggregated attributes in L, and with its value for each aggregate output of γ_L

© North Carolina State University, All Rights Reserved

Compute average GPA of students applying to CS, by city and state

(4) if needed

(5-Π) SELECT city, state, avg(GPA)

(1) FROM Student NC

(2) WHERE EXIST

(select * from Apply
WHERE ID = S.ID
AND major = 'CS')

(3) GROUP by city, state;

© North Carolina State University, All Rights Reserved

for all cases of ≥ 100 applications per major, Return total # of applications and latest application date per major,

(4) ~~(3)~~ select major, count(ID), max(date)
(1) FROM Apply

(2) GROUP BY major
(3) HAVING count(ID) >= 100

© North Carolina State University, All Rights Reserved

Student-NC (ID, name, street Addr, city, state, GPA, SAT)

GROUP BY

Find average SAT score for the students with GPA ≥ 3.7, by city,

HAVING

for cities with ≥ 20 students OR cities whose name starts with R

© North Carolina State University, All Rights Reserved

#5 select city, AVG(SAT)
#1 from StudentNC
#2 where GPA >= 3.7
#3 group by city
#4 having count(*) >= 20
OR city LIKE 'R%'

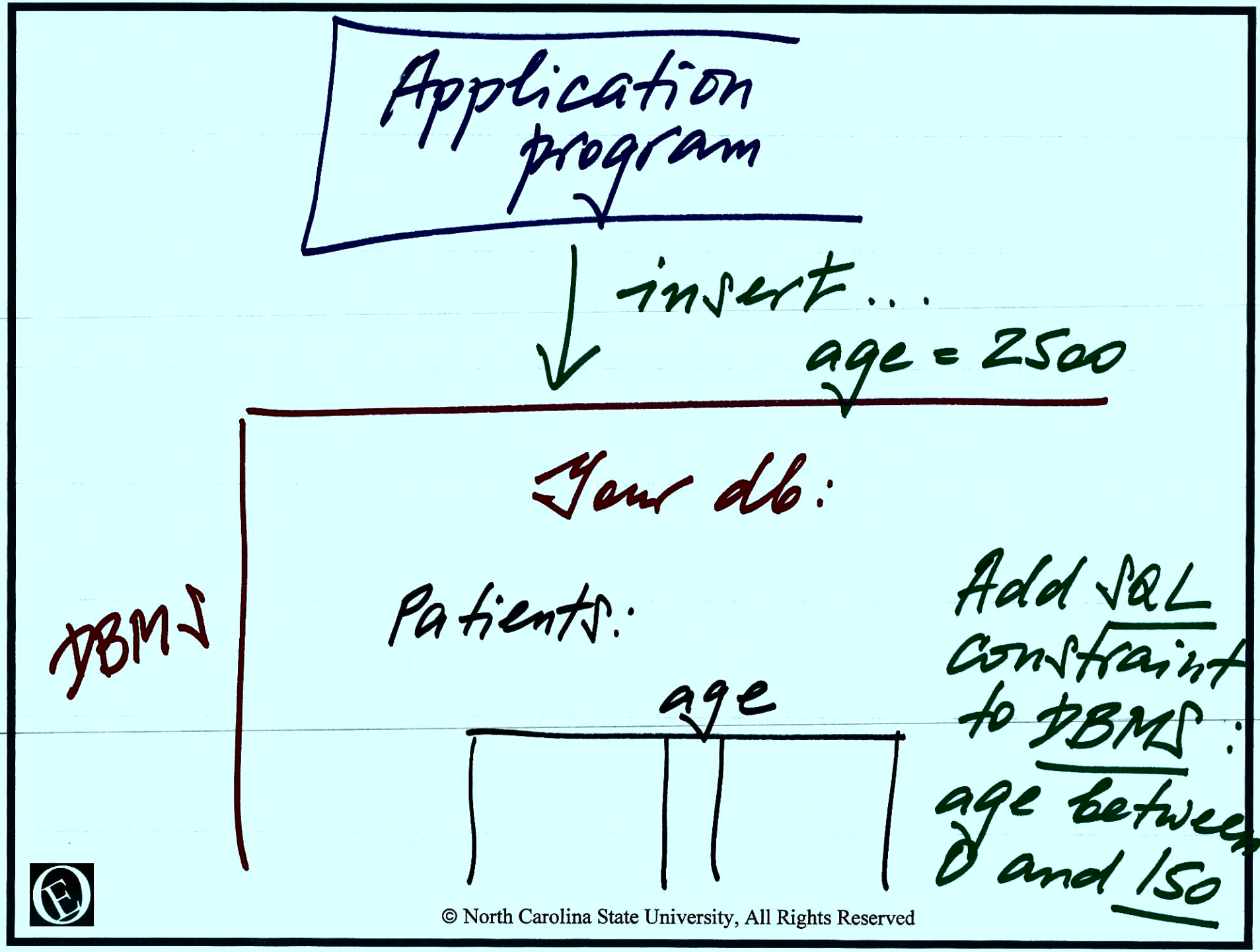
© North Carolina State University, All Rights Reserved

Return IDs and names of students who have applied to ≥ 2 schools:

```
select S.ID, name
from Student S, Apply A
where S.ID = A.ID
group by S.ID, name
having count(*) >= 2
```

distinct location

© North Carolina State University, All Rights Reserved



© North Carolina State University, All Rights Reserved

* Integrity constraints: impose restrictions on the allowable data

* Have seen:
  - keys
  - functional dependencies
  - referential-integrity constraints (foreign-key constraints)

© North Carolina State University, All Rights Reserved

On Student/C, A

- a student may apply to a university no more than once per year
- a student with GPA < 3.0 may only apply to campuses with rank > 4

© North Carolina State University, All Rights Reserved

* Why use:
  - data-entry errors
  - db modifications:
    correctness criteria
  - to enforce consistency across db data

© North Carolina State University, All Rights Reserved

```
create table Student
    (ID integer primary key,
     name char(30) NOT NULL,
     address varchar(30),
     gpa float NOT NULL,
     sat integer,
     UNIQUE (name),
     UNIQUE (address));
```

© North Carolina State University, All Rights Reserved