

Projection column
Aggregate  
↓
↓  
**SELECT** student.name, **MAX**(enrolled.score)  
**FROM** student, enrolled ← Query tables  
**WHERE** student.student\_id = enrolled.student\_id ← Join conditions  
           and student.level = 'senior' ← Query conditions  
**GROUP BY** student.student\_id ← GROUP BY clause  
**HAVING COUNT**(enrolled.course\_id) > 2 ← HAVING clause  
**ORDER BY** student.name ← ORDER BY clause

```


SELECT    student.name, <Aggregate>
FROM      student, enrolled
WHERE      student.student_id = enrolled.student_id
            and <Query Condition>
GROUP BY  < Column Name(s)>
HAVING    <Query Condition>
ORDER BY  <Column Name(s)>

```

Fgg

ff

name	score
Bob	4
Dan	5
Jim	2

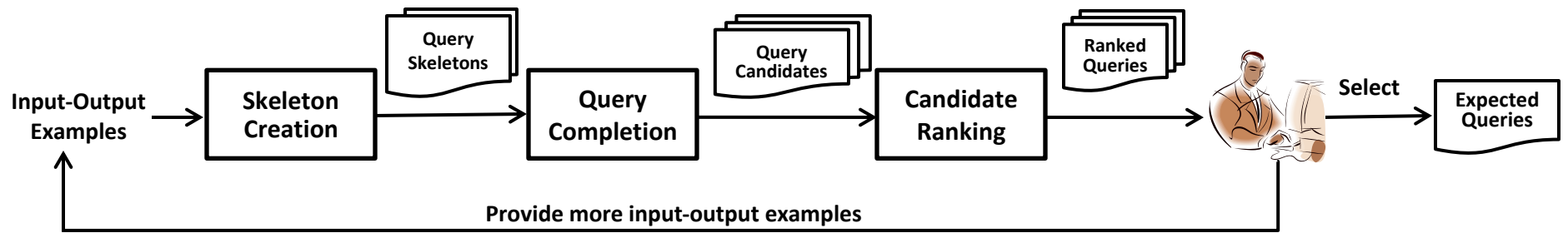


name
Bob
Dan

(a) The input table: student

(b) The output table

1. `select name from student where score > 2`
2. `select name from student where name = 'Bob'`  
`or name = 'Dan'`



student_id	course_id	score	name	level
1	1	4	Adam	senior
1	2	2	Adam	senior
2	1	3	Bob	junior
2	2	2	Bob	junior
2	3	3	Bob	junior
3	2	1	Erin	senior
4	1	4	Rob	junior
4	3	4	Rob	junior
5	2	5	Dan	senior
5	3	2	Dan	senior
5	4	1	Dan	senior
6	2	4	Peter	senior
6	4	5	Peter	senior
7	1	2	Sai	senior
7	3	3	Sai	senior
7	4	4	Sai	senior

(a)

Aggregation Features	
Group by student_id	
COUNT(course_id)	MAX(score)
2	4
2	4
3	3
3	3
3	3
1	1
2	4
2	4
3	5
3	5
3	5
2	5
2	5
3	4
3	4
3	4

COUNT(course\_id) > 2  
&& level = "senior"

student_id	course_id	score	name	level
5	2	5	Dan	senior
5	3	2	Dan	senior
5	4	1	Dan	senior
7	1	2	Sai	senior
7	3	3	Sai	senior
7	4	4	Sai	senior

(b)

Project on column: name,  
and aggregate: MAX(score)

name	max_score
Dan	5
Sai	4

(c)

group by student\_id

student_id	course_id	score	name	level
5	2	5	Dan	senior
5	3	2	Dan	senior
5	4	1	Dan	senior
7	1	2	Sai	senior
7	3	3	Sai	senior
7	4	4	Sai	senior

(d)

(e)

student_id	course_id	score	name	level
2	1	3	Bob	junior
2	2	2	Bob	junior
2	3	3	Bob	junior
5	2	5	Dan	senior
5	3	2	Dan	senior
5	4	1	Dan	senior
7	1	2	Sai	senior
7	3	3	Sai	senior
7	4	4	Sai	senior

# Aggregation Features

An input table

C1	C2
2	4
2	1
2	1
1	1

Group by C1					Group by C2						
COUNT	COUNT	DISTINCT	MIN	MAX	AVG	COUNT	COUNT	DISTINCT	MIN	MAX	AVG
3		2	1	4	2	1		1	2	2	2
3		2	1	4	2	3		2	1	2	5/3
3		2	1	4	2	3		2	1	2	5/3
4		2	1	1	1	3		2	1	2	5/3

Comparison Features

C1 = C2	C1 < C2	C1 > C2
0	1	0
0	0	1
0	0	1
1	0	0

Column1	Column2	Column3	Column 4
101	2001	3020	01-01-11
101	2001	3002	02-01-11
101	2001	3001	03-01-11
102	2002	3002	01-01-11

Column1	Column2	Column 3
20011	2001	200131
20012	2001	200132
20013	2001	200133

Column1	Column 2
20011	Site
20012	Site
20013	Site

101	200131	01-01-11	Site
101	200132	01-01-11	Site
101	200133	01-01-11	Site



```

select min(T1.Column1), T2.Column3,
       min(T1.Column4), min(T3.Column2)
from T1, T2, T3
where T1.Column2 = T2.Column2
      and T2.Column1 = T3.Column1
group by T2.Column3

```

T3 (right)

(b) A SQL query inferred by SQLSythensizer

(c) The output table

student_id	name	level
1	Adam	senior
2	Bob	junior
3	Erin	senior
4	Rob	junior
5	Dan	senior
6	Peter	senior
7	Sai	senior

student_id	course_id	score
1	1	4
1	2	2
2	1	3
2	2	2
2	3	3
3	2	1
4	1	4
4	3	4
5	2	5
5	3	2
5	4	1
6	2	4
6	4	5
7	1	2
7	3	3
7	4	4



name	max_score
Dan	5
Sai	5

```

SELECT student.name, MAX(enrolled.score)
FROM student, enrolled
WHERE student.student_id = enrolled.student_id
      and student.level = 'senior'
GROUP BY student.student_id
HAVING COUNT(enrolled.course_id) > 2

```

(a) Two input tables: student (Left) and enrolled (Right)

(b) A SQL query inferred by SQLSynthesizer

(c) An output table