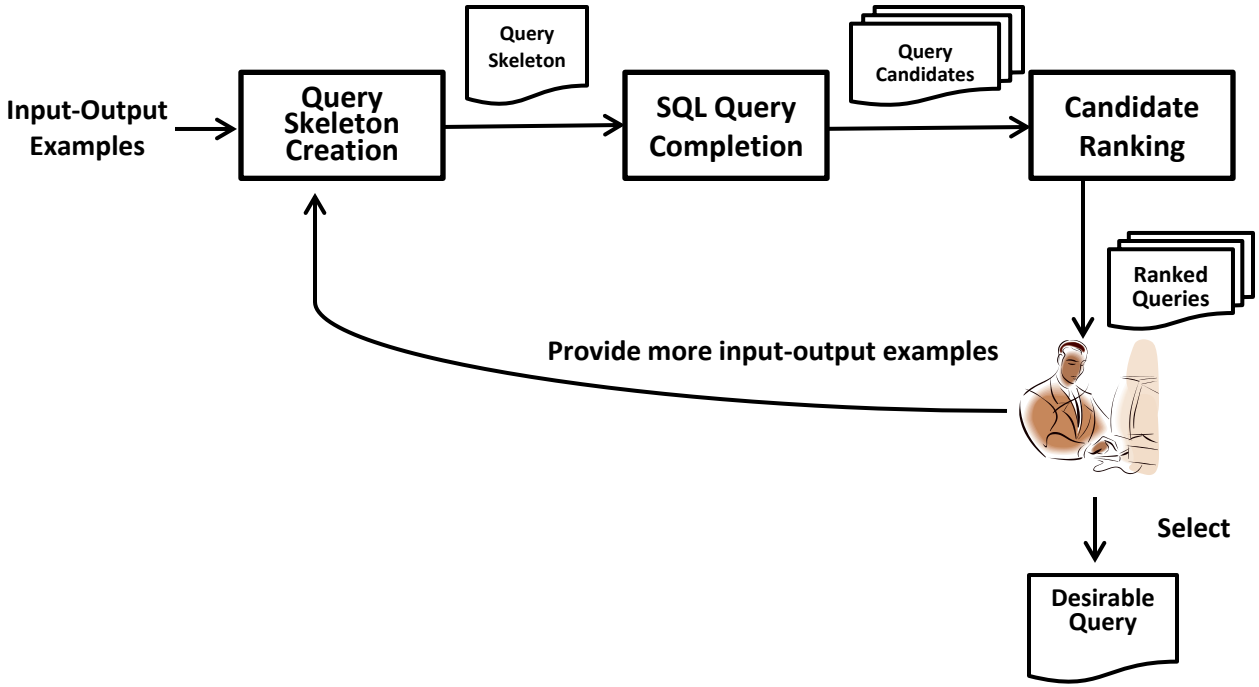
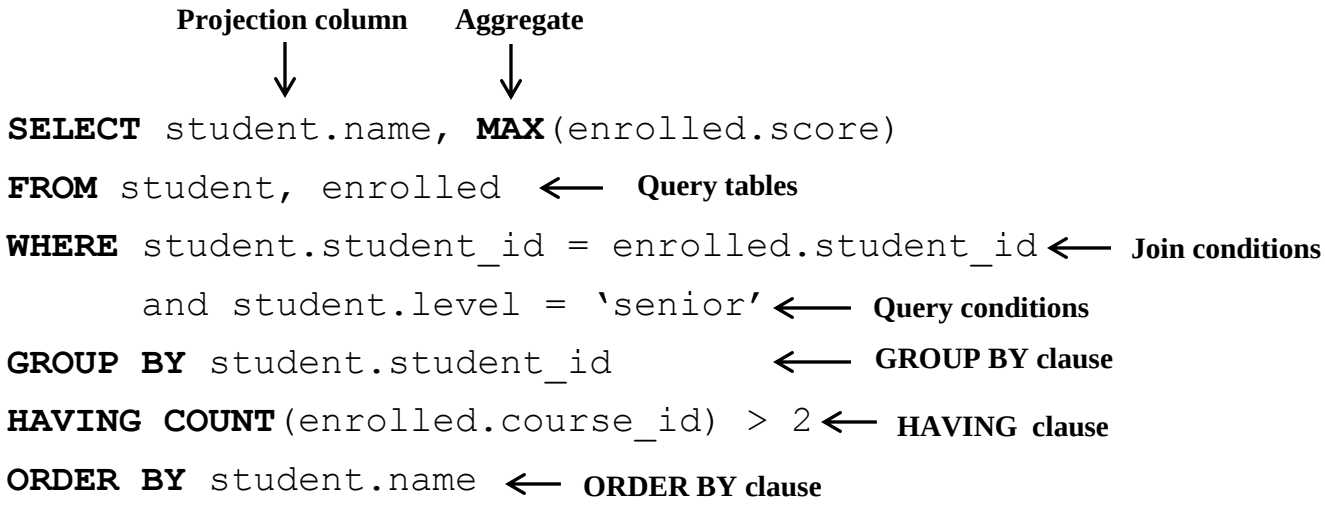

Input-Output Examples
Query Skeleton Creation
Query Skeleton
SQL Query Completion
Query Candidates
Candidate Ranking
Ranked Queries
Provide more input-output examples
Select
Desirable Query


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)>
```

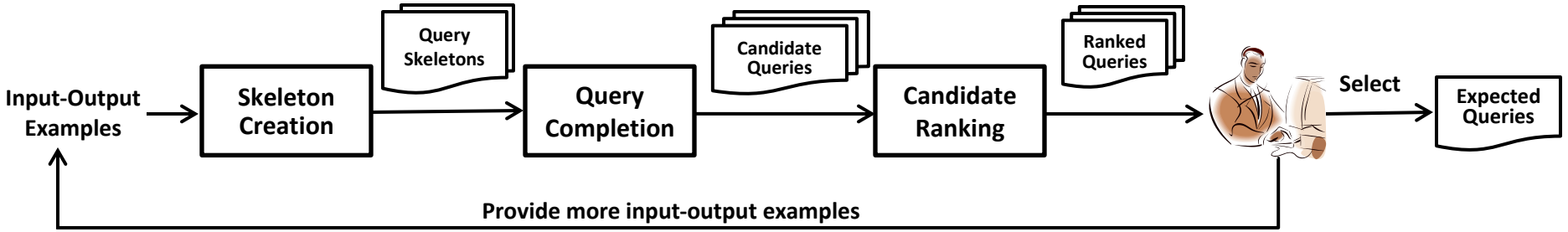
name	score
Bob	4
Dan	5
Jim	2

(a) An input table: student

name
Bob
Dan

(b) An output table

1. `SELECT name FROM student WHERE score > 2`
2. `SELECT name FROM student WHERE name = 'Bob' or name = 'Dan'`


Input-Output Examples
Skeleton Creation
Query Skeletons
Query Completion
Candidate Queries
Candidate Ranking
Ranked Queries
Select
Expected Queries
Provide more input-output examples

The table created by joining table **student** with table **enrolled** on column **student_id**

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	5	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	5	Sai	senior

(b)

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

name	max_score
Dan	5
Sai	5

(c)

An input table

C1	C2
2	4
2	1
2	1
1	1

Aggregation Features

Group by **C1**						Group by **C2**					
COUNT(**C2**)	COUNT (DISTINCT **C2**)	MIN(**C2**)	MAX(**C2**)	SUM(**C2**)	AVG(**C2**)	COUNT(**C1**)	COUNT (DISTINCT **C1**)	MIN(**C1**)	MAX(**C1**)	SUM(**C1**)	AVG(**C1**)
3	2	1	4	6	2	1	1	2	2	2	2
3	2	1	4	6	2	3	2	1	2	5	5/3
3	2	1	4	6	2	3	2	1	2	5	5/3
1	1	1	1	1	1	3	2	1	2	5	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

```
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
```

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

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	5

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

```
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
```

(b) A SQL query inferred by SQLSynthesizer

name	max_score
Dan	5
Sai	5

(c) An output table