Subqueries

A Restriction on Aggregate Functions

- What if we also wanted the *name* of the max-capacity room?

```
SELECT MAX(capacity)
FROM Room
WHERE name LIKE 'CAS%';
```

Room

id	*name*	*capacity*
1000	CAS Tsai	500
2000	CAS BigRoom	100
3000	EDU Lecture Hall	100
4000	CAS 315	40
5000	CAS 314	80
6000	CAS 226	50
7000	MCS 205	30

A Restriction on Aggregate Functions

- What if we also wanted the *name* of the max-capacity room?

```
SELECT name, MAX(capacity)
FROM Room
WHERE name LIKE 'CAS%';
```

This does *not* work in standard SQL!

Room

id	*name*	*capacity*
1000	CAS Tsai	500
2000	CAS BigRoom	100
3000	EDU Lecture Hall	100
4000	CAS 315	40
5000	CAS 314	80
6000	CAS 226	50
7000	MCS 205	30

WHERE

id	*name*	*capacity*
1000	CAS Tsai	500
2000	CAS BigRoom	100
4000	CAS 315	40
5000	CAS 314	80
6000	CAS 226	50

SELECT name

name
CAS Tsai
CAS BigRoom
CAS 315
CAS 314

A Restriction on Aggregate Functions

- What if we also wanted the *name* of the max-capacity room?

```
SELECT name, MAX(capacity)
FROM Room
WHERE name LIKE 'CAS%';
```

This does *not* work in standard SQL!

Room

id	*name*	*capacity*
1000	CAS Tsai	500
2000	CAS BigRoom	100
3000	EDU Lecture Hall	100
4000	CAS 315	40
5000	CAS 314	80
6000	CAS 226	50
7000	MCS 205	30

WHERE

id	*name*	*capacity*
1000	CAS Tsai	500
2000	CAS BigRoom	100
4000	CAS 315	40
5000	CAS 314	80
6000	CAS 226	50

SELECT name

name
CAS Tsai
CAS BigRoom
CAS 315
CAS 314

MAX

MAX(capacity)
500

error!

A Restriction on Aggregate Functions (cont.)

- What if we also wanted the *name* of the max-capacity room?

```
SELECT name, MAX(capacity)
FROM Room
WHERE name LIKE 'CAS%';
```

This does *not* work in standard SQL!

- In general, a SELECT clause *cannot* combine:
  - an aggregate function
  - a column name that is on its own (and is not being operated on by an aggregate function)

Subqueries

- A *subquery* allows us to use the result of one query in the evaluation of another query.

- We can use a subquery to solve the previous problem:

```
SELECT name, capacity
FROM Room
WHERE name LIKE 'CAS%'
  AND capacity = (SELECT MAX(capacity)
                  FROM Room
                  WHERE name LIKE 'CAS%');
```

the subquery

Note Carefully!

```
SELECT name, capacity
FROM Room
WHERE name LIKE 'CAS%'
  AND capacity = (SELECT MAX(capacity)
                  FROM Room
                  WHERE name LIKE 'CAS%');
```

the subquery

Note Carefully!

```
SELECT name, capacity
FROM Room
WHERE name LIKE 'CAS%'
  AND capacity = (SELECT MAX(capacity)
                  FROM Room
                  WHERE name LIKE 'CAS%');
```

the subquery

- if we remove the condition from the subquery, might not get the largest capacity in CAS
- if we remove the condition from the outer query, might also get rooms from other buildings

Subqueries and Set Membership

- Subqueries can be used to test for *set membership* in conjunction with the `IN` and `NOT IN` operators.
  - example: find all students who are not enrolled in CS 105

```
SELECT name
FROM Student
WHERE id NOT IN (SELECT student_id
                 FROM Enrolled
                 WHERE course_name = 'CS 105');
```

Enrolled

student_id	*course_name*	*credit_status*
12345678	CS 105	ugrad
25252525	CS 111	ugrad
45678900	CS 460	grad
33566891	CS 105	non-credit
45678900	CS 510	grad

subquery 

student_id
12345678
33566891

Subqueries and Set Membership

- Subqueries can be used to test for *set membership* in conjunction with the `IN` and `NOT IN` operators.
  - example: find all students who are not enrolled in CS 105

```
SELECT name
FROM Student
WHERE id NOT IN (SELECT student_id
                 FROM Enrolled
                 WHERE course_name = 'CS 105');
```

Enrolled

student_id	*course_name*	*credit_status*
12345678	CS 105	ugrad
25252525	CS 111	ugrad
45678900	CS 460	grad
33566891	CS 105	non-credit
45678900	CS 510	grad

Student

id	*name*
12345678	Jill Jones
25252525	Alan Turing
33566891	Audrey Chu
45678900	Jose Delgado
66666666	Count Dracula

subquery

student_id
12345678
33566891

name
Alan Turing
Jose Delgado
Count Dracula

```
mysql> select *from employee;
+-----+--------+------+----------+
| eno | ename  | dno  | job      |
+-----+--------+------+----------+
| 11  | a      |    1 | manager  |
| 12  | b      |    1 | manager  |
| 13  | c      |    2 | clerk    |
+-----+--------+------+----------+
3 rows in set (0.00 sec)
```

```
mysql> select *from employee;
+-----+-------+------+
| eno | ename | dno  |
+-----+-------+------+
| 11  | a     |    1 |
| 12  | b     |    1 |
| 13  | c     |    2 |
+-----+-------+------+
3 rows in set (0.00 sec)

mysql> select *from deptm;
+-----+-------+
| dno | dname |
+-----+-------+
|   1 | acc   |
|   2 | rec   |
|   3 | sales |
+-----+-------+
3 rows in set (0.00 sec)
```

```
mysql> select *from deptm d where exists(select e.eno from employee e where e.dno=d.dno);
+-----+-------+
| dno | dname |
+-----+-------+
|   1 | acc   |
|   2 | rec   |
+-----+-------+
2 rows in set (0.00 sec)

mysql> select *from deptm d where not exists(select e.eno from employee e where e.dno=d.dno);
+-----+-------+
| dno | dname |
+-----+-------+
|   3 | sales |
+-----+-------+
1 row in set (0.00 sec)
```

Query!- List down the employees whose salary is greater than avg of their department;

Emp (Eid, Name, sal, dep)

1	A	2k	CSE
2	B	3k	EC
3	C	3K	CSE
4	D	4K	EC

Query!- List down the employees whose salary is greater than avg of their department;

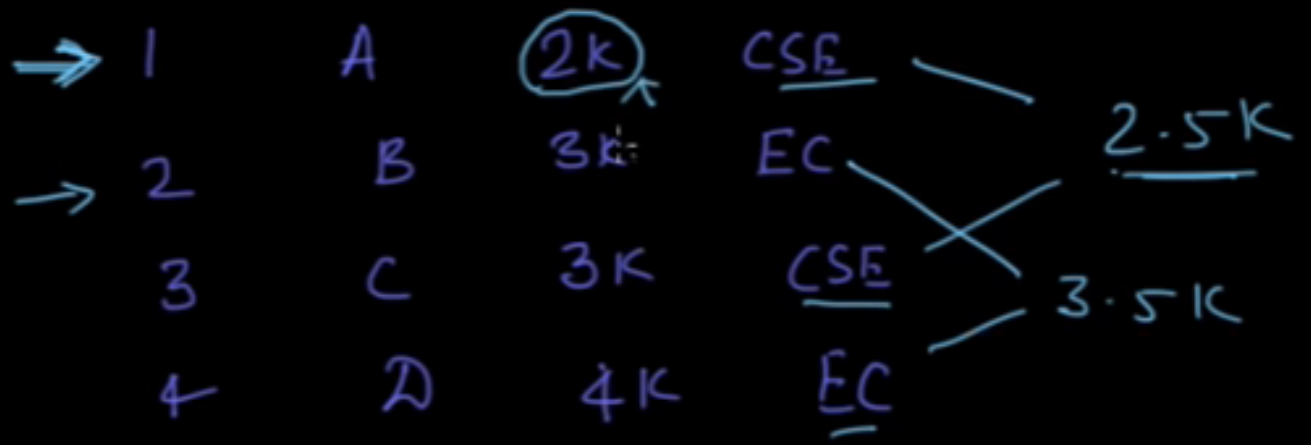
Emp (Eid, Name, sal, dep)

→ 1	A	2k	CSE
2	B	3k	EC
3	C	3K	CSE
4	D	4K	EC

CSE → 2.5K

Query!- List down the employees whose salary is greater than avg of their department ;

Emp (Eid, Name, sal, dep)



Emp (Eid , Name , sal , dep)

→	1	A	2k	CSE
→	2	B	3k	EC
	3	C	3K	CSE
	4	D	4K	EC

2.5K

3.5K

```
SELECT Eid , Name FROM Emp as E
WHERE Sal > ( SELECT Avg(sal)
              FROM Emp
              WHERE dep = E.dep) ;
```

Performance of correlated subquery :-

It performs better if only a few records are retrieved by Outer query.

OR

Inner query returns only a small records.

Otherwise, VIEWS or JOINS will be more efficient.

```
mysql> select ename, eno from employee where job=(select ename from employee where eno='12');
Empty set (0.00 sec)

mysql> select ename, eno from employee where job=(select job from employee where eno='12');
+-------+-----+
| ename | eno |
+-------+-----+
| a     | 11  |
| b     | 12  |
+-------+-----+
2 rows in set (0.00 sec)

mysql> select ename, eno from employee where eno!=12 and job=(select job from employee where eno='12');
+-------+-----+
| ename | eno |
+-------+-----+
| a     | 11  |
+-------+-----+
1 row in set (0.00 sec)
```