

# More SQL

January 30, 2020

Data Science CSCI 1951A

Brown University

Instructor: Ellie Pavlick

HTAs: Josh Levin, Diane Mutako, Sol Zitter

# Announcements

- People with overrides—Friday EOD to use codes, then we will give them to someone else. No new codes going out.
- Project mixer—tomorrow at 4, 4th floor
- Sign the collab policy!

# Outline

- Last time: SQL for creating/manipulating data tables
- Today: SQL for querying databases
  - Keywords
  - NULLs
  - Execution Order, Nested Queries, Optimization

# Basic Template

```
SELECT <attribute list>  
FROM <table list>  
[ WHERE <condition> ]  
[ GROUP BY <attribute list> ]  
[ HAVING <condition> ]  
[ ORDER BY <attribute list> ];
```

# TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT <attribute list>  
FROM <table list>  
WHERE <condition>;
```

```
SELECT *  
FROM TWEET  
WHERE Text = "hey"
```



ID	Time	Text
389472	2019-01-01 12:34:56	hey
127890	2019-01-04 17:30:07	hey

# TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT <attribute list>  
FROM <table list>  
WHERE <condition>;
```

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```



ID	Time
389472	2019-01-01 12:34:56
127890	2019-01-04 17:30:07

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet,
      Author
WHERE ID = Tweet
```

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Person
FROM Tweet,
      Author
WHERE ID = Tweet
```



## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
389472	2019-01-01	hey	s	389472
123794	2019-01-01	lol	d	123794
596208	2019-01-02	:-D	j	596208
782138	2019-01-04	1951A 4 lyfe	d	782138
127890	2019-01-04	hey	d	127890
173902	2019-01-05	i <3 1951A	j	173902
893110	2019-01-06	i <3 1951A	s	893110



## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Person
FROM Tweet,
      Author
WHERE ID = Tweet
```

"Join Condition"



## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
389472	2019-01-01	hey	s	389472
123794	2019-01-01	lol	d	123794
596208	2019-01-02	:-D	j	596208
782138	2019-01-04	1951A 4 lyfe	d	782138
127890	2019-01-04	hey	d	127890
173902	2019-01-05	i <3 1951A	j	173902
893110	2019-01-06	i <3 1951A	s	893110

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Person
FROM Tweet,
      Author
```

No condition →  
cross product

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
389472	2019-01-01	hey	s	389472
389472	2019-01-01	hey	d	123794
389472	2019-01-01	hey	j	596208
389472	2019-01-01	hey	d	782138
389472	2019-01-01	hey	d	127890
389472	2019-01-01	hey	j	173902
389472	2019-01-01	hey	s	893110
123794	2019-01-01	lol	s	389472
...	10	...	...	...

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Person
FROM Tweet,
      Author
WHERE ID = Tweet
```

*\*actually, even with join condition, will do cross product first*

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
389472	2019-01-01	hey	s	389472
123794	2019-01-01	lol	d	123794
596208	2019-01-02	:-D	j	596208
782138	2019-01-04	1951A 4 lyfe	d	782138
127890	2019-01-04	hey	d	127890
173902	2019-01-05	i <3 1951A	j	173902
893110	2019-01-06	i <3 1951A	s	893110

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.Tweet
```

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

aliasing

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.ID
```

## AUTHOR

Person	ID
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

*aliasing (to avoid ambiguity)*

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.Tweet
      AND a.Person = "d"
```

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.Tweet
      AND a.Person = "d"
```

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
389472	2019-01-01	hey	s	389472
123794	2019-01-01	lol	d	123794
596208	2019-01-02	:-D	j	596208
782138	2019-01-04	1951A 4 lyfe	d	782138
127890	2019-01-04	hey	d	127890
173902	2019-01-05	i <3 1951A	j	173902
893110	2019-01-06	i <3 1951A	s	893110

## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.Tweet
      AND a.Person = "d"
```



## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

ID	Time	Text	Person	Tweet
123794	2019-01-01 12:34:57	lol	d	123794
782138	2019-01-04 15:04:57	1951A 4 lyfe	d	782138
127890	2019-01-04 17:30:07	hey	d	127890



## TWEET

ID	Time	Text
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
596208	2019-01-02 3:14:15	:-D
782138	2019-01-04 15:04:57	1951A 4 lyfe
127890	2019-01-04 17:30:07	hey
173902	2019-01-05 3:34:18	i <3 1951A
893110	2019-01-06 12:21:53	i <3 1951A

## AUTHOR

Person	Tweet
s	389472
d	123794
j	596208
d	782138
d	127890
j	173902
s	893110

```
SELECT ID, Text
FROM Tweet AS t,
      Author AS a
WHERE t.ID = a.Tweet
      AND a.Person = "d"
```



ID	Text
123794	lol
782138	1951A 4 lyfe
127890	hey

# Clicker Question!

# Clicker Question!

PERSON

Handle	Name
s	Sol
d	Diane
j	Josh

RETWEET

Person	Tweet
s	1
s	2
d	1

Find names of people who have retweeted.

# Clicker Question!

PERSON

Handle	Name
s	Sol
d	Diane
j	Josh

RETWEET

Person	Tweet
s	1
s	2
d	1

Pst. Hint →

Person
Sol
Sol
Diane

# Clicker Question!

PERSON

Handle	Name
s	Sol
d	Diane
j	Josh

RETWEET

Person	Tweet
s	1
s	2
d	1

**(a)**

```
SELECT Name
FROM PERSON AS p,
     RETWEET AS r
WHERE r.Person = p.Name
```

**(b)**

```
SELECT *
FROM PERSON AS p,
     RETWEET AS r
WHERE r.Person = p.Handle
```

**(c)**

```
SELECT Name
FROM PERSON AS p,
     RETWEET AS r
WHERE r.Person = p.Handle
```

# Clicker Question!

PERSON

Handle	Name
s	Sol
d	Diane
j	Josh

RETWEET

Person	Tweet
s	1
s	2
d	1

**(a)**

```
SELECT Name
FROM PERSON AS p,
      RETWEET AS r
WHERE r.Person = p.Name
```

**(b)**

```
SELECT *
FROM PERSON AS p,
      RETWEET AS r
WHERE r.Person = p.Handle
```

**(c)**

```
SELECT Name
FROM PERSON AS p,
      RETWEET AS r
WHERE r.Person = p.Handle
```

# JOINS

```
SELECT ID, Text  
FROM TWEET, AUTHOR  
WHERE ID = Tweet AND Person = "d"
```

# JOINS

```
SELECT ID, Text
FROM TWEET, AUTHOR
WHERE ID = Tweet AND Person = "d"
```

=

```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR ON ID = Tweet)
WHERE Person = "d"
```



# JOINS

```
SELECT ID, Text
FROM TWEET, AUTHOR
WHERE ID = Tweet AND Person = "d"
```

=

```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR ON ID = Tweet)
WHERE Person = "d"
```

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR
      ON ID = Tweet)
```

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	672109		
		782138	1951A 4 lyfe

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
<b>d</b>	<b>672109</b>

```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR
      ON ID = Tweet)
```

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
<b>d</b>	<b>672109</b>		
		782138	1951A 4 lyfe

# JOINS

TWEET

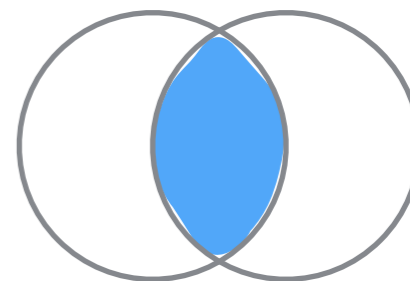
ID	Text
389472	hey
596208	:-D
<b>782138</b>	<b>1951A 4 lyfe</b>
173902	i <3 1951A
893110	i <3 1951A

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
<b>d</b>	<b>672109</b>

```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR
      ON ID = Tweet)
```

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
<b>d</b>	<b>672109</b>		
		<b>782138</b>	<b>1951A 4 lyfe</b>



Inner Join

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

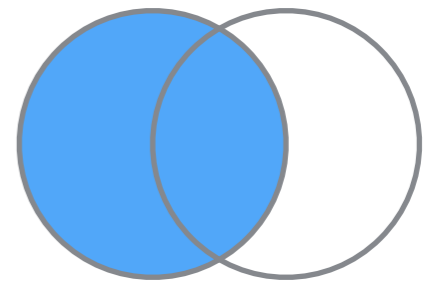
```
SELECT ID, Text
FROM (TWEET JOIN AUTHOR
      ON ID = Tweet)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	<del>672109</del>		
		<del>782138</del>	<del>1951A 4 lyfe</del>

# JOINS



Left Outer Join

TWEET

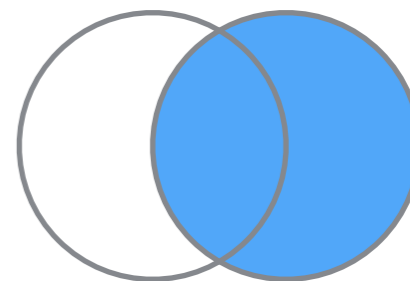
ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET LEFT OUTER JOIN AUTHOR
      ON ID = Tweet)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	<del>672109</del>		
NULL	NULL	782138	1951A 4 lyfe



# JOINS

Right Outer Join

TWEET

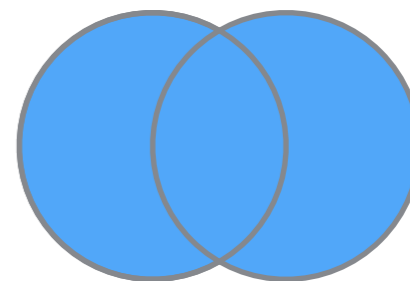
ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET RIGHT OUTER JOIN AUTHOR
      ON ID = Tweet)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	672109	NULL	NULL
		782138	1951A 4 lyfe



# JOINS

Full Outer Join

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET FULL OUTER JOIN AUTHOR
      ON ID = Tweet)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	Tweet	ID	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	672109	NULL	NULL
NULL	NULL	782138	1951A 4 lyfe



# JOINS

Natural Join

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text  
FROM (TWEET JOIN AUTHOR)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

assumes condition is  
ALL PAIRS of attributes with  
matching names

# JOINS

Natural Join

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text  
FROM (TWEET JOIN AUTHOR)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

if no matches, forms full  
cross product

# JOINS

Natural Join

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET AS t(tweetid, text) JOIN
      AUTHOR AS a(person, tweetid))
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	tweetid	tweetid	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A
d	672109	NULL	NULL
NULL	NULL	782138	1951A 4 lyfe

Natural (Inner)  
Join

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET AS t(tweetid, text) JOIN
      AUTHOR AS a(person, tweetid))
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	tweetid	tweetid	Text
s	389472	389472	hey
j	596208	596208	:-D
j	173902	173902	i <3 1951A
s	893110	893110	i <3 1951A

Natural (Inner)  
Join

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET AS t(tweetid, text) JOIN
      AUTHOR AS a(person, tweetid)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

Person	tweetid	Text
s	389472	hey
j	596208	:-D
j	173902	i <3 1951A
s	893110	i <3 1951A

Natural (Inner)  
Join

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET AS t(tweetid, foo) JOIN
      AUTHOR AS a(foo, tweetid)
```

AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

foo	tweetid	foo
s	389472	hey
j	596208	:-D
j	173902	i <3 1951A
s	893110	i <3 1951A

Natural (Inner)  
Join

# JOINS

TWEET

ID	Text
389472	hey
596208	:-D
782138	1951A 4 lyfe
173902	i <3 1951A
893110	i <3 1951A

```
SELECT ID, Text
FROM (TWEET AS t(tweetid, foo) JOIN
      AUTHOR AS a(foo, tweetid)
```



AUTHOR

Person	Tweet
s	389472
j	596208
j	173902
s	893110
d	672109

# Clicker Question! (x2)



# Clicker Question!

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name, Course  
FROM (STUDENT LEFT OUTER JOIN GRADES ON ID = Student)
```

Name	Course
Diane	32
Sol	1951A
NULL	32

**(a)**

Name	Course
Diane	32
Sol	1951A
Josh	NULL
Karlly	NULL
Mounika	NULL

**(b)**

# Clicker Question!

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name, Course  
FROM (STUDENT LEFT OUTER JOIN GRADES ON ID = Student)
```

Name	Course
Diane	32
Sol	1951A
NULL	32

(a)

Name	Course
Diane	32
Sol	1951A
Josh	NULL
Karlly	NULL
Mounika	NULL

(b)

# Clicker Question!

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

Target →

Name	Course
Diane	32
Sol	1951A
NULL	32

(a)

```
SELECT Name, Course  
FROM (STUDENT RIGHT OUTER JOIN GRADES ON ID = Student)
```

(b)

```
SELECT Name, Course  
FROM (STUDENT NATURAL JOIN GRADES ON ID = Student)
```

# Clicker Question!

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

Target →

Name	Course
Diane	32
Sol	1951A
NULL	32

(a)

```
SELECT Name, Course  
FROM (STUDENT RIGHT OUTER JOIN GRADES ON ID = Student)
```

(b)

```
SELECT Name, Course  
FROM (STUDENT NATURAL JOIN GRADES ON ID = Student)
```

let me **Google** that for you

Google Search

I'm Feeling Lucky

Enable javascript to use LMGTFY.

---

[About](#) [Privacy](#) [Contact](#) [@LMGTFY](#) [Shop Shirts](#) [Shop Stickers](#) [iPhone App](#) [Live Stream](#)

And now...a laundry list  
of keywords...

# ORDER BY

TWEET

ID	Time	Text
782138	2019-01-04 15:04:57	1951A 4 lyfe
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
127890	2019-01-04 17:30:07	hey
893110	2019-01-06 12:21:53	i <3 1951A
596208	2019-01-02 3:14:15	:-D
173902	2019-01-05 3:34:18	i <3 1951A

```
SELECT Text
FROM Tweet
ORDER BY Time
```

Text
hey
lol
:-D
1951A 4 lyfe
hey
i <3 1951A
i <3 1951A

# ORDER BY

TWEET

ID	Time	Text
782138	2019-01-04 15:04:57	1951A 4 lyfe
389472	2019-01-01 12:34:56	hey
123794	2019-01-01 12:34:57	lol
127890	2019-01-04 17:30:07	hey
893110	2019-01-06 12:21:53	i <3 1951A
596208	2019-01-02 3:14:15	:-D
173902	2019-01-05 3:34:18	i <3 1951A

```
SELECT Text  
FROM Tweet  
ORDER BY ID
```

Text
lol
hey
i <3 1951A
hey
:-D
1951A 4 lyfe
i <3 1951A

# GROUP BY

TWEET

ID	Likes	Text
782138	1,000	1951A 4 lyfe
389472	10	hey
123794	100	lol
127890	0	hey
893110	8,000,000	i <3 1951A
596208	1	:-D
173902	1,000,000,000	i <3 1951A

```
SELECT Text,  
Count(*), AVG(Likes)  
FROM Tweet  
GROUP BY Text
```

Text	Count(*)	AVG(Likes)
lol	1	100
hey	2	5
i <3 1951A	2	504,000,000
:-D	1	1
1951A 4 lyfe	1	1,000



# GROUP BY

TWEET

ID	Likes	Text
782138	1,000	1951A 4 lyfe
389472	10	hey
123794	100	lol
127890	0	hey
893110	8,000,000	i <3 1951A
596208	1	:-D
173902	1,000,000,000	i <3 1951A

```
SELECT Text,  
Count(*), AVG(Likes)  
FROM Tweet  
GROUP BY Text
```

Text	Count(*)	AVG(Likes)
lol	1	100
hey	2	5
i <3 1951A	2	504,000,000
:-D	1	1
1951A 4 lyfe	1	1,000

SUM, MIN, MAX,  
COUNT, AVG

# HAVING

TWEET

ID	Likes	Text
782138	1,000	1951A 4 lyfe
389472	10	hey
123794	100	lol
127890	0	hey
893110	8,000,000	i <3 1951A
596208	1	:-D
173902	1,000,000,000	i <3 1951A

```
SELECT Text,  
Count(*), AVG(Likes)  
FROM Tweet  
GROUP BY Text  
HAVING COUNT(*) > 1
```

Text	Count(*)	AVG(Likes)
hey	2	5
i <3 1951A	2	504,000,000

# HAVING

TWEET

ID	Likes	Text
782138	1,000	1951A 4 lyfe
389472	10	hey
123794	100	lol
127890	0	hey
893110	8,000,000	i <3 1951A
596208	1	:-D
173902	1,000,000,000	i <3 1951A

```
SELECT Text,  
Count(*), AVG(Likes)  
FROM Tweet  
GROUP BY Text  
HAVING COUNT(*) > 1
```

Text	Count(*)	AVG(Likes)
hey	2	5
i <3 1951A	2	504,000,000

similar behavior to "WHERE", but only  
used with aggregations/GROUP BYs

# LIKE

TWEET

ID	Likes	Text
782138	1,000	1951A 4 lyfe
389472	10	hey
123794	100	lol
127890	0	hey
893110	8,000,000	i <3 1951A
596208	1	:-D
173902	1,000,000,000	i <3 1951A

```
SELECT Text, Count(*),  
AVG(Likes)  
FROM Tweet  
WHERE Text LIKE '%1951A%'  
GROUP BY Text
```

Text	Count(*)	AVG(Likes)
1951A 4 lyfe	1	1,000
i <3 1951A	2	504,000,000

# IN

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name
FROM STUDENT
WHERE ID IN
    (SELECT Student
     FROM GRADES
     WHERE Course = 1951A
    )
```

Find names of  
students in 1951A

# IN

"Subquery"  
(More later, get excited)

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name
FROM STUDENT
WHERE ID IN
  (SELECT Student
   FROM GRADES
   WHERE Course = 1951A
  )
```

Find names of  
students in 1951A

# IN

Returns "bag"  
of student IDs


STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name
FROM STUDENT
WHERE ID IN
  (SELECT Student
   FROM GRADES
   WHERE Course = 1951A
  )
```



Find names of  
students in 1951A

# IN

Returns True if  
ID is in that bag

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	32	A
2	1951A	A
6	32	A

```
SELECT Name
FROM STUDENT
WHERE ID IN
    (SELECT Student
     FROM GRADES
     WHERE Course = 1951A
    )
```

Find names of  
students in 1951A



# ALL/ANY

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade >= ALL
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

What is the **highest** grade in 1951A?

# ALL/ANY

Returns True if condition holds  
for all tuples in bag

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
AND Grade >= ALL
(SELECT Grade
FROM GRADES
WHERE Course = 1951A
)
```

What is the **highest**  
grade in 1951A?

# ALL/ANY

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade > ANY
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

???

# ALL/ANY

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade > ANY
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

Return all grades  
**except the lowest one.**

# ALL/ANY

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade > NOT ANY
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

Return the **lowest** grade.

# ALL/ANY


STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade >= ALL
      (SELECT Grade
       FROM GRADES
       WHERE Course = 1951A
      )
```



Grade
3.5
3.5

# DISTINCT

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT DISTINCT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade >= ALL
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

↓

Grade
3.5

# DISTINCT

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT DISTINCT Grade
FROM GRADES
WHERE Course = "1951A"
      AND Grade >= ALL
      (SELECT Grade
      FROM GRADES
      WHERE Course = 1951A
      )
```

↓

Grade
3.5

Set operations (Union, Intersection, etc.) remove duplicates by default.



# EXISTS

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT NAME
FROM STUDENT s
WHERE NOT EXISTS
  (SELECT *
   FROM GRADES
   WHERE Course = 1951A
   AND Student = s.ID
  )
```

???

# EXISTS

True as long as bag is not empty

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karlly
5	Mounika

GRADES

Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT NAME
FROM STUDENT s
WHERE NOT EXISTS
  (SELECT *
   FROM GRADES
   WHERE Course = 1951A
   AND Student = s.ID
  )
```

???

# EXISTS

STUDENT

ID	Name
1	Diane
2	Sol
3	Josh
4	Karly
5	Mounika

GRADES


Student	Course	Grade
1	1951A	3.5
2	1951A	3.5
6	1951A	2.8

```
SELECT NAME
FROM STUDENT s
WHERE NOT EXISTS
  (SELECT *
   FROM GRADES
   WHERE Course = 1951A
   AND Student = s.ID
  )
```

Students who are  
not in 1951A



# Outline

- Last time: SQL for creating/manipulating data tables
- Today: SQL for querying databases
  - Keywords 
  - NULLs
  - Execution Order, Nested Queries, Optimization

# NULL!

- Black hole! NULL is NULL is NULL and there is no coming back from it...
- If an operand is NULL, the result is NULL:
  - $\text{NULL} + 1 = \text{NULL}$
  - $\text{NULL} * 0 = \text{NULL}$
- Comparisons: All comparisons that involve a null value, evaluate to unknown
  - $\text{NULL} = \text{NULL} \rightarrow \text{Unknown}$
  - $\text{NULL} \neq \text{NULL} \rightarrow \text{Unknown}$
  - $\text{NULL} < 13 \rightarrow \text{Unknown}$
  - $\text{NULL} > \text{NULL} \rightarrow \text{Unknown}$

# NULL!

p	q	p OR q	p AND q	p = q
TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE

# NULL!

p	q	p OR q	p AND q	p = q
TRUE	TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
TRUE	UNK	TRUE	UNK	UNK
FALSE	UNK	UNK	FALSE	UNK
UNK	TRUE	TRUE	UNK	UNK
UNK	FALSE	UNK	FALSE	UNK
UNK	UNK	UNK	UNK	UNK



# NULL!

WHERE: Only tuples which evaluate to true are part of the query result. (I.e. **unknown and false treated equivalently.**)

TWEET		
ID	Text	Likes
389472	NULL	100
123794	NULL	3
596208	:-D	NULL
782138	1951A 4 lyfe	NULL
173902	i <3 1951A	19
893110	i <3 1951A	7539

```
SELECT COUNT(*)  
FROM TWEET  
WHERE Likes > 10
```



Count(*)
3

# NULL!

GROUP BY: If NULL exists, then there is a group for NULL.

TWEET		
ID	Text	Likes
389472	NULL	100
123794	NULL	3
596208	:-D	NULL
782138	1951A 4 lyfe	NULL
173902	i <3 1951A	19
893110	i <3 1951A	7539

```
SELECT Text, COUNT(*)  
FROM TWEET  
GROUP BY Text
```



Text	Count(*)
NULL	2
:-D	1
1951A 4 lyfe	1
i <3 1951A	2

# NULL!

For predicates with NULL, use IS (as opposed to “=“)

ID	Text	Likes
389472	NULL	100
123794	NULL	3
596208	:-D	NULL
782138	1951A 4 lyfe	NULL
173902	i <3 1951A	19
893110	i <3 1951A	7539

```
SELECT Text ID  
FROM TWEET  
WHERE Text = NULL
```



ID

# NULL!

For predicates with NULL, use IS (as opposed to “=“)

TWEET		
ID	Text	Likes
389472	NULL	100
123794	NULL	3
596208	:-D	NULL
782138	1951A 4 lyfe	NULL
173902	i <3 1951A	19
893110	i <3 1951A	7539

```
SELECT Text ID
FROM TWEET
WHERE Text IS NULL
```



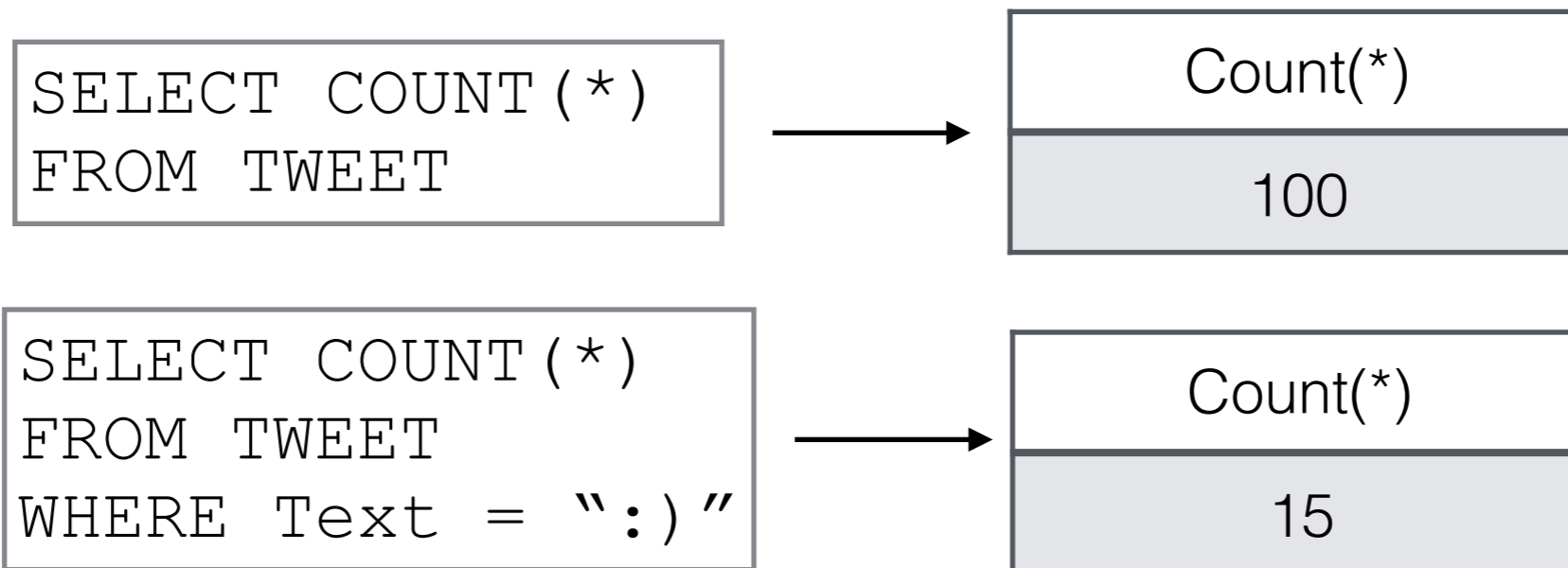
ID
389472
123794

# NULL!

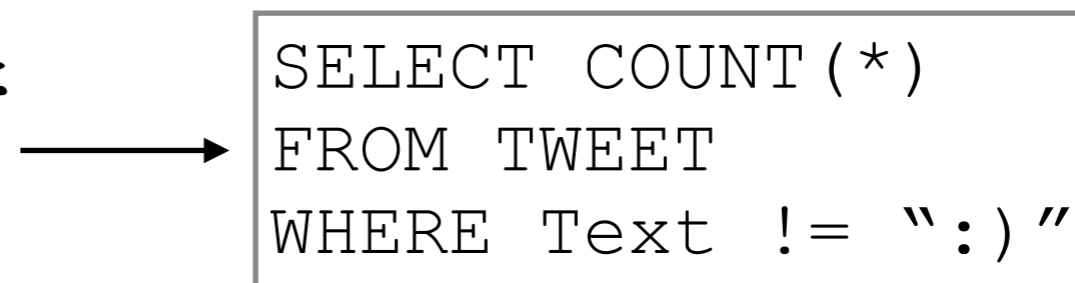
- `count (att)`: NULL is ignored
- `sum (att)`: NULL is ignored
- `avg (att)`: results from SUM and COUNT
- `min (att)` and `max (att)`: NULL is ignored
- Exception! If NULL is the only value in the column, then `sum/avg/min/max` all return "NULL"

# Clicker Question! (x2)

# Clicker Question!



**What will be the result of  
this query?**



**(a)**

Count(*)
100

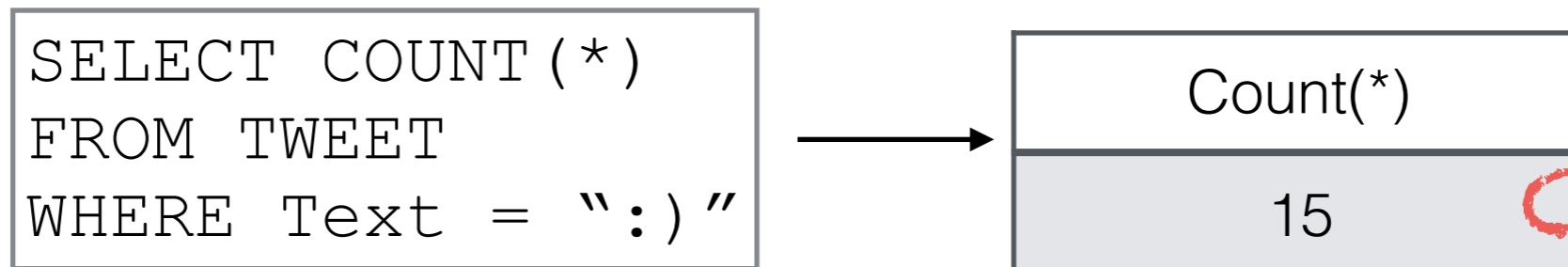
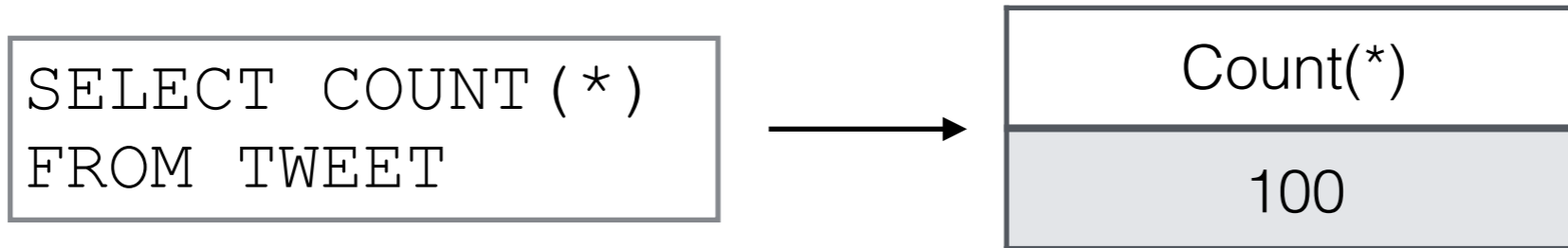
**(b)**

Count(*)
85 79

**(c)**

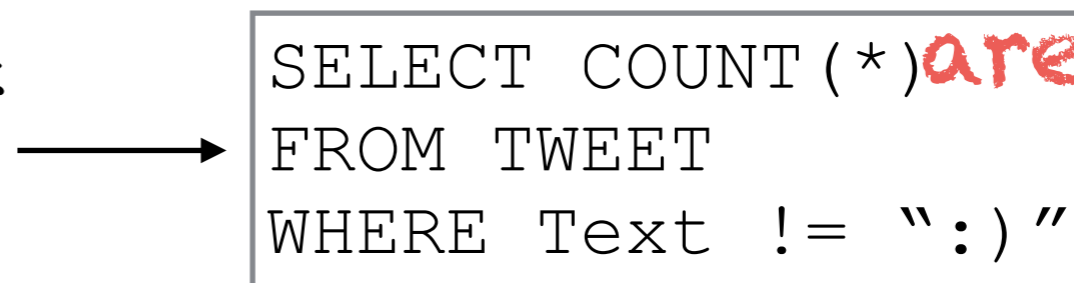
**I...don't...know...**

# Clicker Question!



Can't say  
how many

What will be the result of  
this query?



are NULL

(a)

Count(*)
100

(b)

Count(*)
85 80

(c)

I...don't...know...



# Clicker Question!

RUNNERS

ID	Name
1	Diane
2	Sol
3	Josh
4	Jon

RACES

Event_ID	Event	Winner_ID
1	PVD Marathon	2
2	PVD Half	3
3	PVD 2 yard jump	2
4	Race4NULL: Raising Awareness	NULL

What will be the result of the below query?

```
SELECT COUNT (*)  
FROM RUNNERS  
WHERE ID NOT IN SELECT (Winner_ID FROM RACES)
```

(a)

(b)

(c)

Count(*)
0

Count(*)
1

Count(*)
2

# Clicker Question!

RUNNERS

ID	Name
1	Diane
2	Sol
3	Josh
4	Jon

RACES

Event_ID	Event	Winner_ID
1	PVD Marathon	2
2	PVD Half	3
3	PVD 2 yard jump	2
4	Race4NULL: Raising Awareness	NULL

What will be the result of the below query?

```
SELECT COUNT (*)  
FROM RUNNERS  
WHERE ID NOT IN (SELECT Winner_ID FROM RACES)
```

*ID NOT IN (2,3,NULL) is the same  
as ID!=2 AND ID!=3 and ID!=NULL*

**(a)**

Count(*)
0

**(b)**



Count(*)
1

**(c)**

Count(*)
2



# Outline

- Last time: SQL for creating/manipulating data tables
- Today: SQL for querying databases
  - Keywords 
  - NULLs 
  - Execution Order, Nested Queries, Optimization

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Text
389472	hey

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Text
389472	hey

SQL

```
SELECT ID, Text  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

FROM

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Text
389472	hey

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

WHERE  
|  
FROM

# Execution Order

TWEET

ID	Time	Text
389472	12:34:5	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Text
389472	hey

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

```
SELECT
|
WHERE
|
FROM
```



# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Text
389472	hey

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

A query can have  
multiple  
"equivalent" trees

FROM

# Execution Order

TWEET

ID	Time	Text
389472	12:34:5	hey
123794	12:34:5	lol
596208	3:14:15	:-D
782138	15:04:5	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:5	i <3 1951A



ID	Text
389472	hey

A query can have multiple "equivalent" trees

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

```
SELECT
|
FROM
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:5	hey
123794	12:34:5	lol
596208	3:14:15	:-D
782138	15:04:5	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:5	i <3 1951A



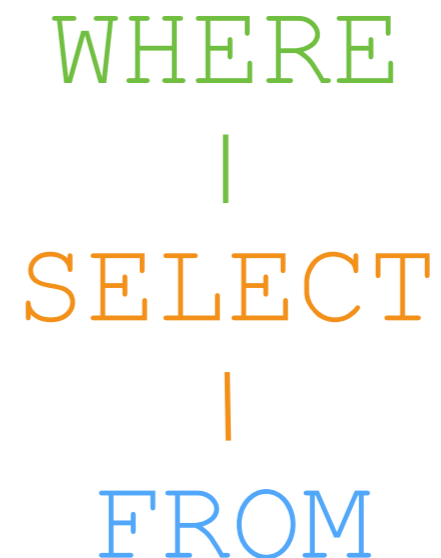
ID	Text
389472	hey

A query can have multiple "equivalent" trees

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree



# Clicker Question!

# Clicker Question!

## Which is better?

**(a)** `WHERE ( SELECT ( FROM ) )`

**(b)** `SELECT ( WHERE ( FROM ) )`

**(c) I...don't...know...it depends**

# Clicker Question!

## Which is better?

(a) `WHERE ( SELECT ( FROM ) )`

(b) `SELECT ( WHERE ( FROM ) )`

(c) **I...don't...know...it depends**

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A

SQL

```
SELECT ID, Text
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A

SQL

```
SELECT ID, Time
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```



# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A

SQL

```
SELECT ID, Time
FROM TWEET
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time
389472	12:34:56
123794	12:34:57
596208	3:14:15
782138	15:04:57
173902	3:34:18
893110	12:21:53

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE ( SELECT ( FROM ) )
```

```
SELECT ( WHERE ( FROM ) )
```

WHAT IS THIS I DON'T EVEN



# Order

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

ID	Time	Text
389472	12:34:56	
123794	12:34:57	
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time
389472	12:34:56
123794	12:34:57
596208	3:14:15
782138	15:04:57
173902	3:34:18
893110	12:21:53

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time	Text
389472	12:34:56	hey

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

TWEET

ID	Time	Text
389472	12:34:56	hey
123794	12:34:57	lol
596208	3:14:15	:-D
782138	15:04:57	1951A 4 lyfe
173902	3:34:18	i <3 1951A
893110	12:21:53	i <3 1951A



ID	Time
389472	12:34:56

SQL

```
SELECT ID, Time  
FROM TWEET  
WHERE Text = "hey"
```

Execution Tree

```
WHERE (SELECT (FROM) )
```

```
SELECT (WHERE (FROM) )
```

# Execution Order

“Canonical Execution Order”

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

```
SELECT  
|  
WHERE  
|  
FROM
```

```
SELECT (WHERE (FROM) )
```



# Clicker Question!



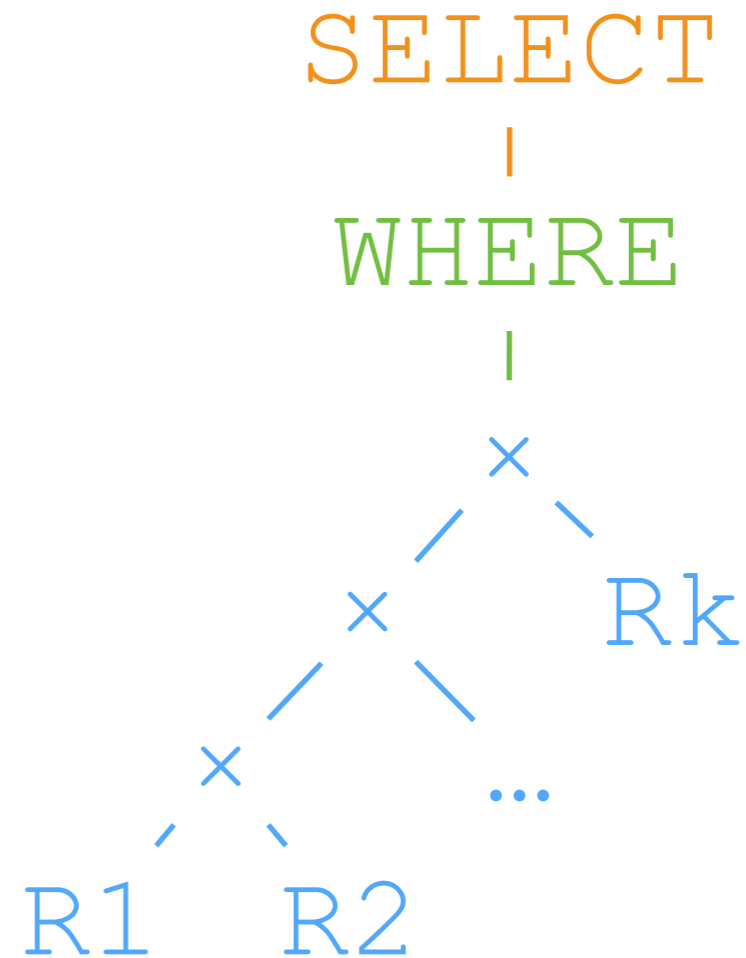
# Clicker Question!

## How much memory do I need?

say each  $R$  has  
 $O(m)$  tuples

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

- (a)**  $O(m^k)$
- (b)  $O(m \times k)$
- (c)  $O(m + k)$
- (d)  $O(m^{k-n})$



# Clicker Question!

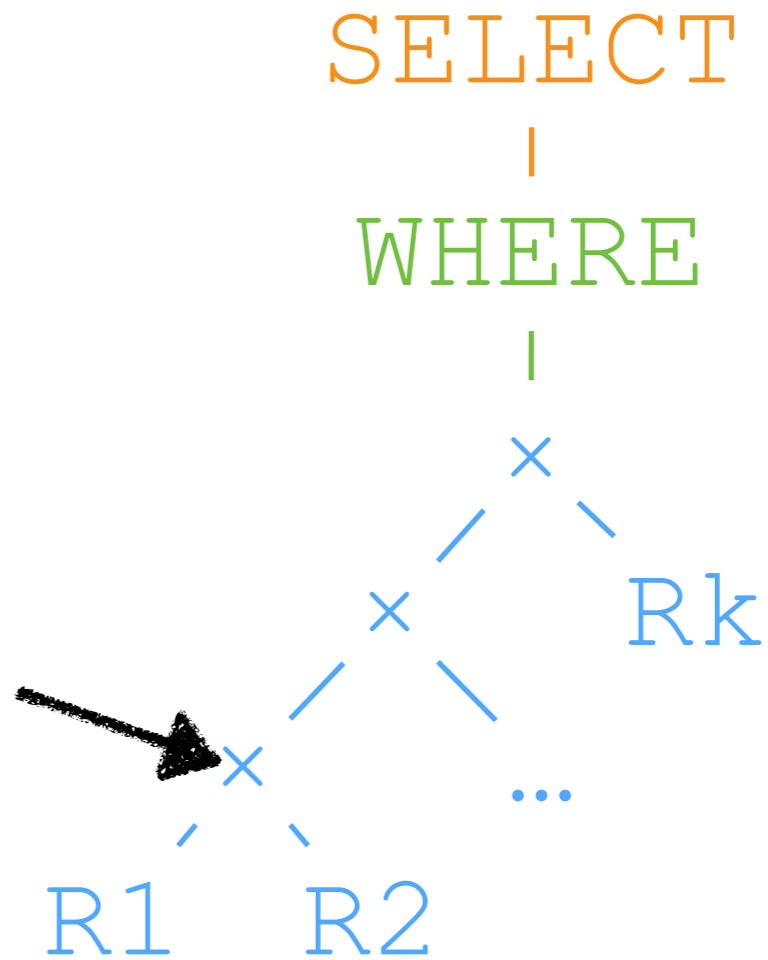
## How much memory do I need?

say each R has  
 $O(m)$  tuples

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

- (a)**  $O(m^k)$
- (b)  $O(m \times k)$
- (c)  $O(m + k)$
- (d)  $O(m^{k-n})$

$m \times m$



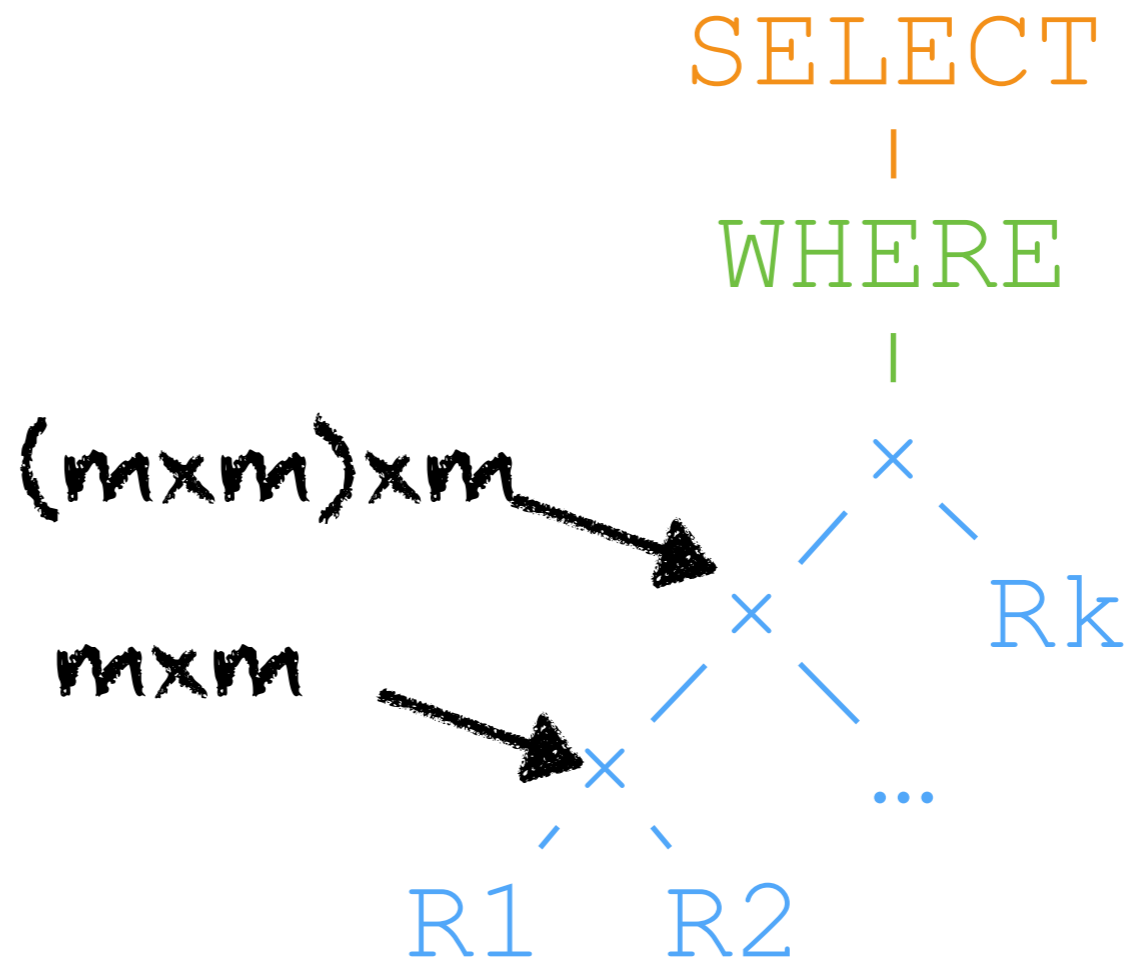
# Clicker Question!

## How much memory do I need?

say each R has  
 $O(m)$  tuples

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

- (a)**  $O(m^k)$
- (b)  $O(m \times k)$
- (c)  $O(m + k)$
- (d)  $O(m^{k-n})$



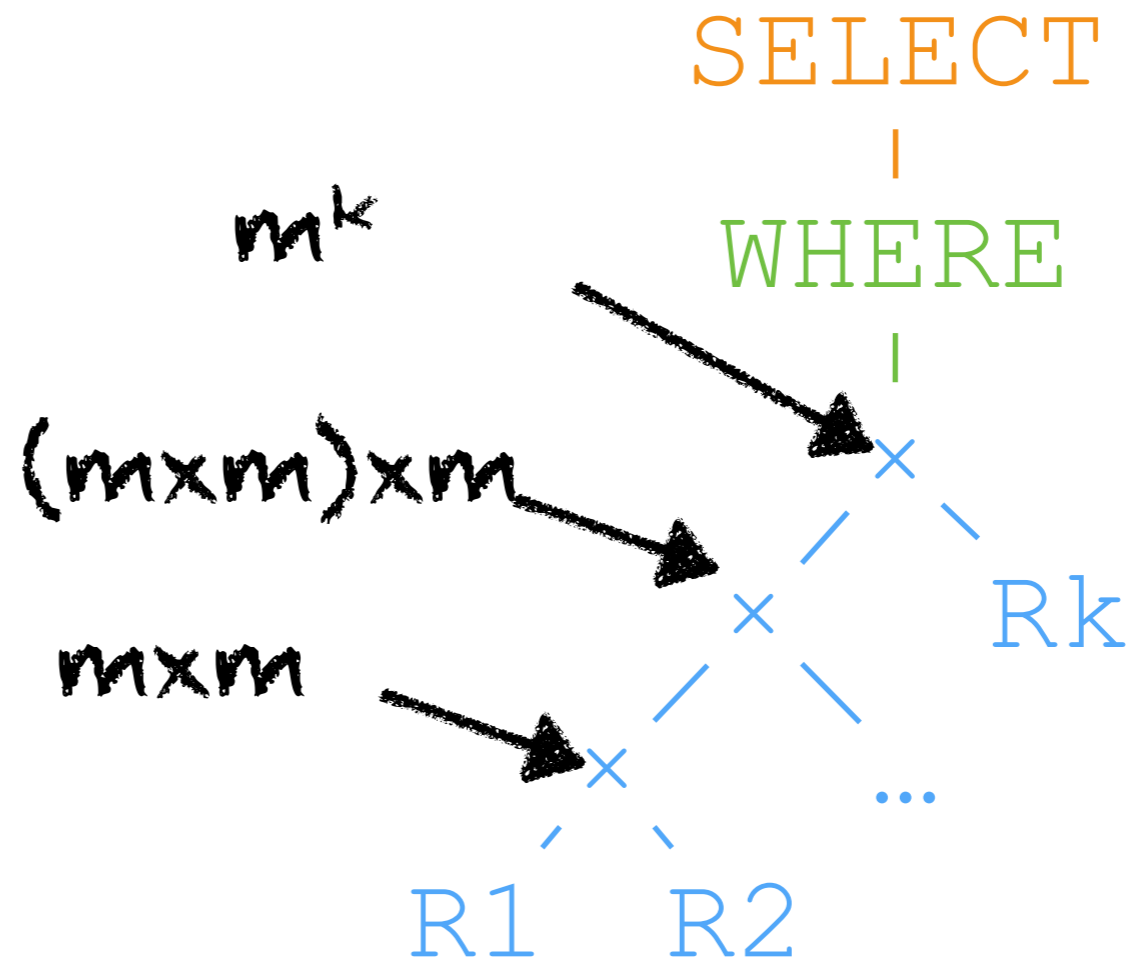
# Clicker Question!

## How much memory do I need?

say each  $R$  has  
 $O(m)$  tuples

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

- (a)**  $O(m^k)$
- (b)  $O(m \times k)$
- (c)  $O(m + k)$
- (d)  $O(m^{k-n})$



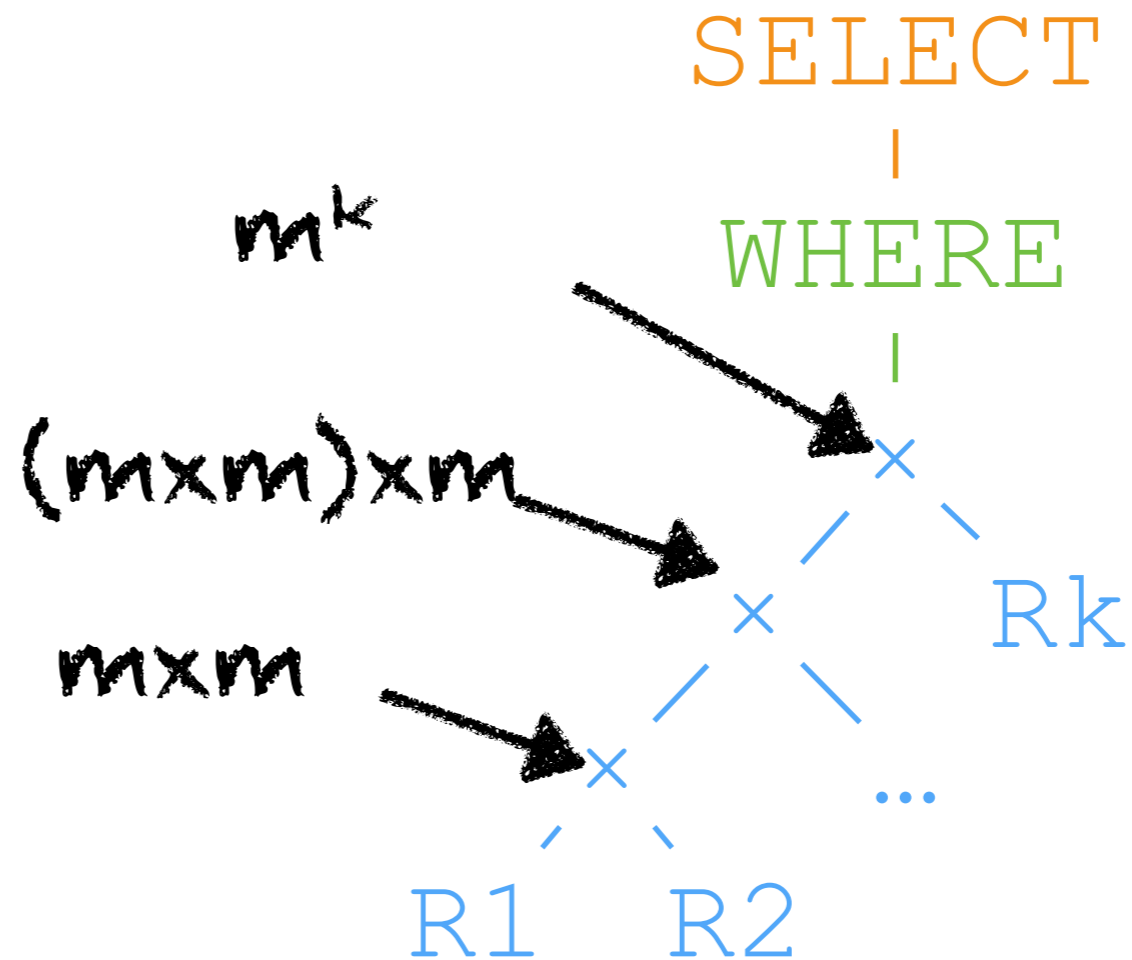
# Clicker Question!

## How much memory do I need?

say each R has  
 $O(m)$  tuples

```
SELECT A1...An  
FROM R1...Rk  
WHERE P
```

- (a)**  $O(m^k)$
- (b)  $O(m \times k)$
- (c)  $O(m + k)$
- (d)  $O(m^{k-n})$



$m = 1000, k = 3$

$\rightarrow$  1 billion tuples

# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

```
SELECT TWEET.Time
      |
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person = "BarackObama")
      |
      X
     / \
    TWEET AUTHOR
```

“Canonical Execution Order” (FROM WHERE SELECT)



# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

```
SELECT TWEET.Time
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person = "BarackObama")
```

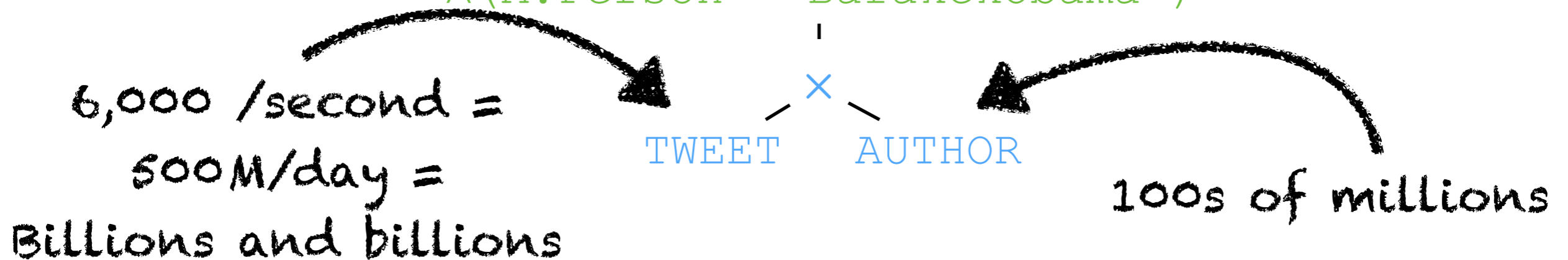


6,000 /second =  
500M/day =  
Billions and billions

# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

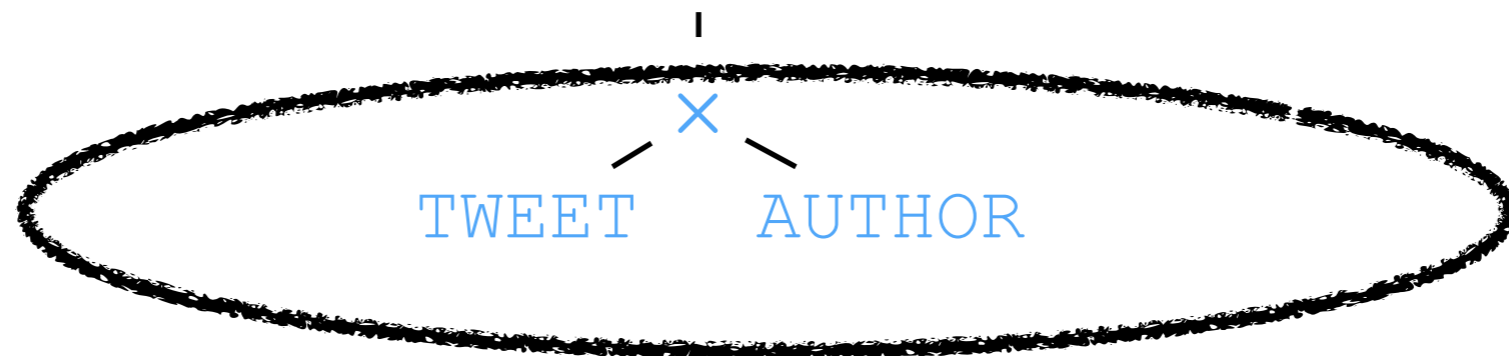
```
SELECT TWEET.Time
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person = "BarackObama")
```



# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

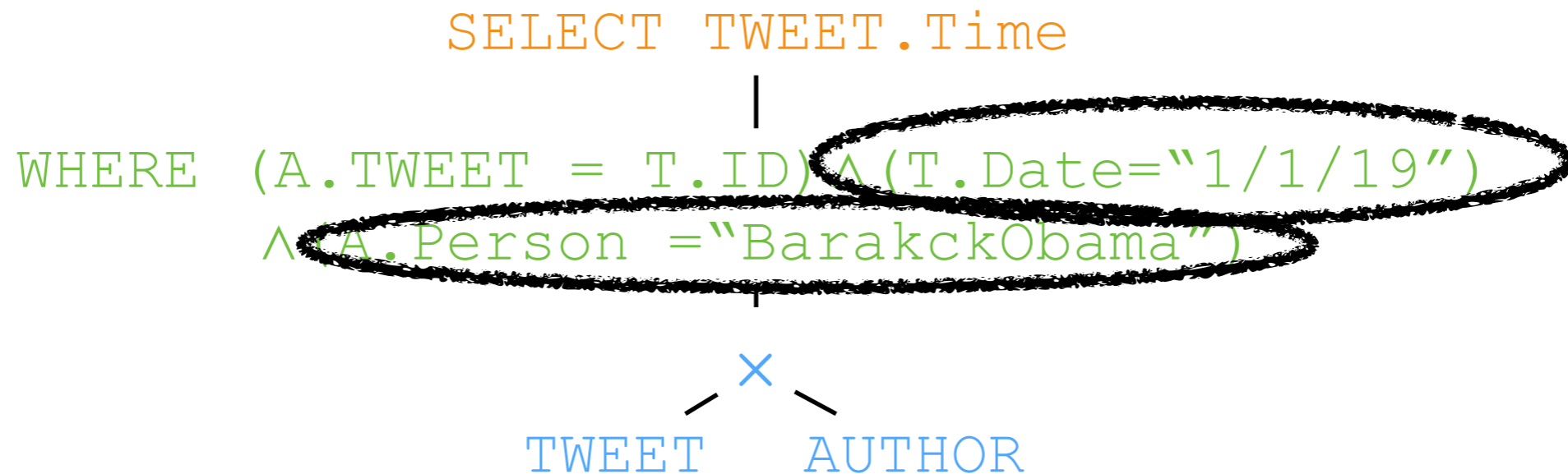
```
SELECT TWEET.Time
      |
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person = "BarackObama")
```



*O(really \*\*\*\*ing big)*

# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```



*O(kind of tiny)*

# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

```
SELECT TWEET.Time
      |
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person="BarackObama")
      |
      x
     / \
    TWEET AUTHOR
```

Thoughts??

*O(kind of tiny)*

# Execution Order

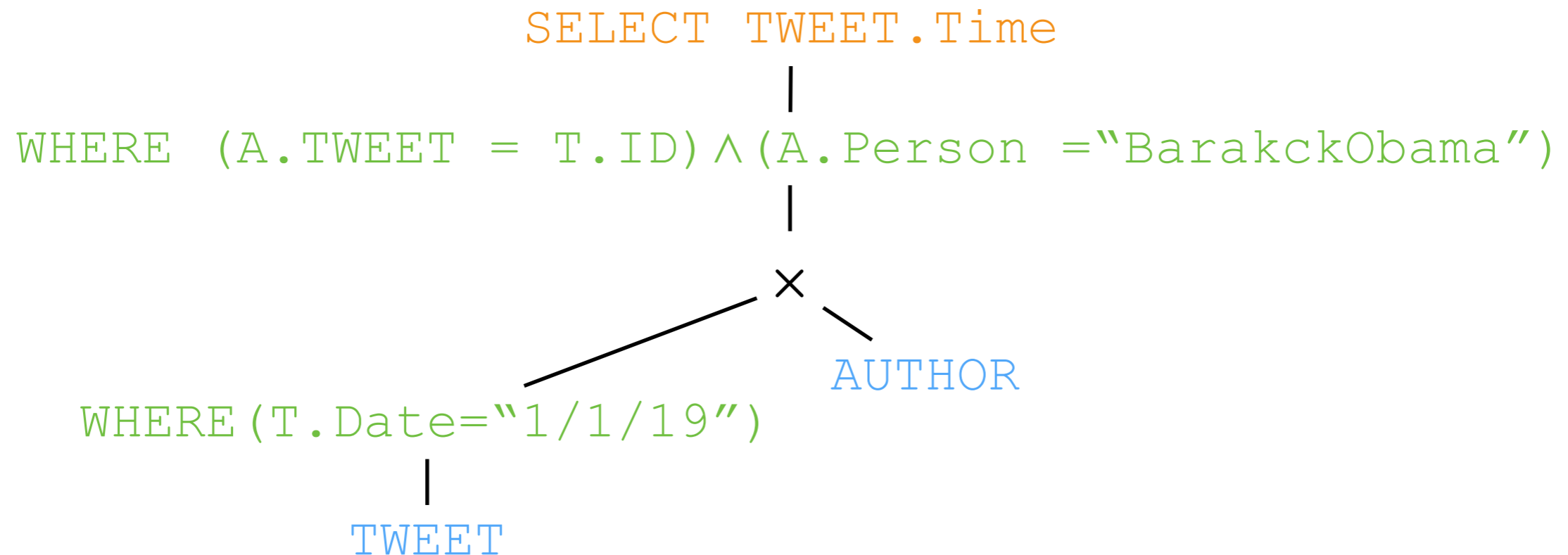
```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```

```
SELECT TWEET.Time
|
WHERE (A.TWEET = T.ID) ^ (T.Date="1/1/19")
      ^ (A.Person = "BarackObama")
|
X
/ \
TWEET AUTHOR
```

**Refactor!**

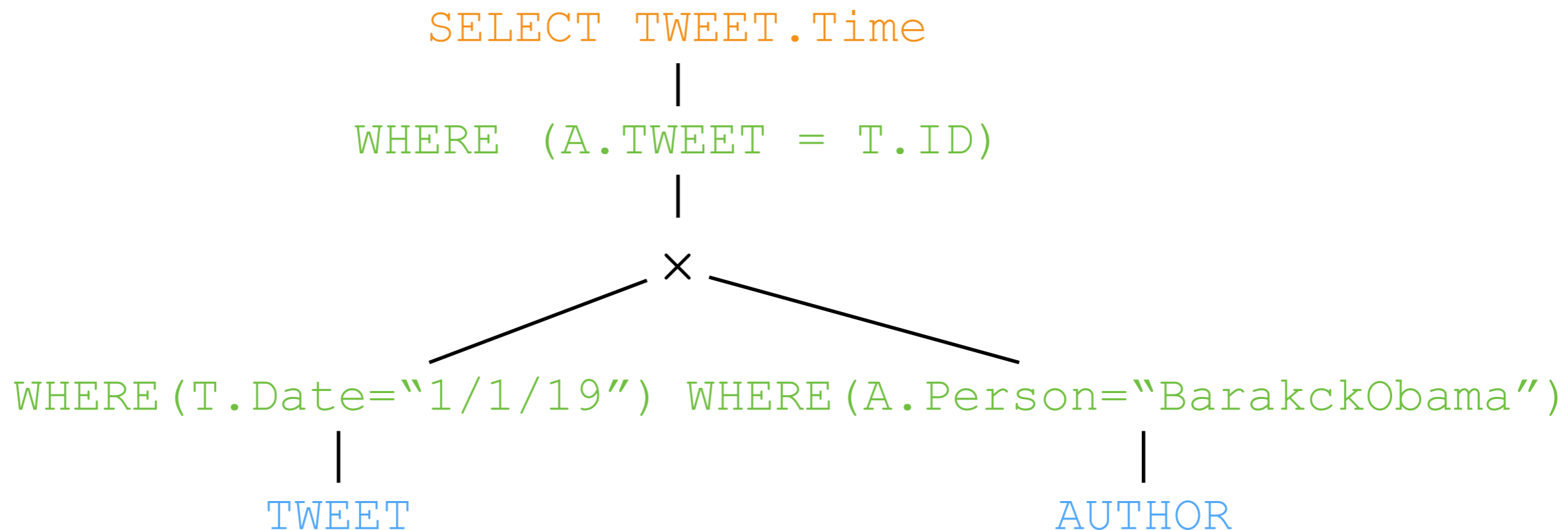
# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```



# Execution Order

```
SELECT TWEET.Time
FROM TWEET, AUTHOR
WHERE AUTHOR.TWEET = TWEET.ID
      and TWEET.Date == '01/01/2019'
      and AUTHOR.Person = "BarackObama"
```





# Clicker Question!

# Clicker Question! (Demand?)

**Optimize this.** Find grades of students taking 1951A ahead of schedule

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

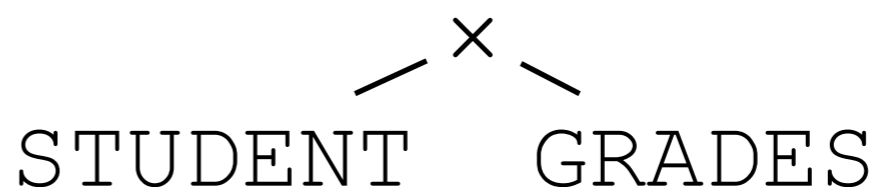
```
SELECT Grade
FROM STUDENT, GRADES
WHERE STUDENT.ID = GRADES.Student
      and GRADES.Course == '1951A'
      and STUDENT.Year < GRADES.Tgt_Yr
```

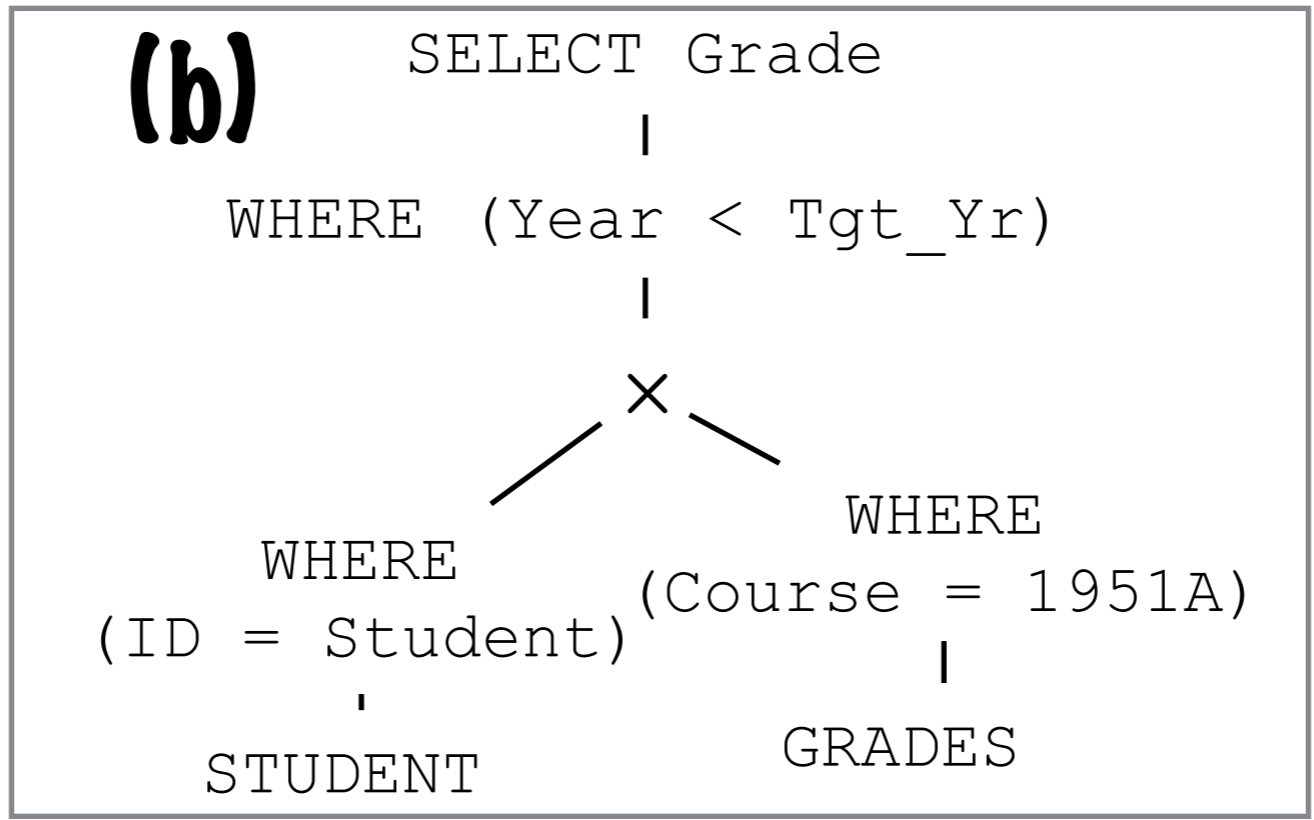
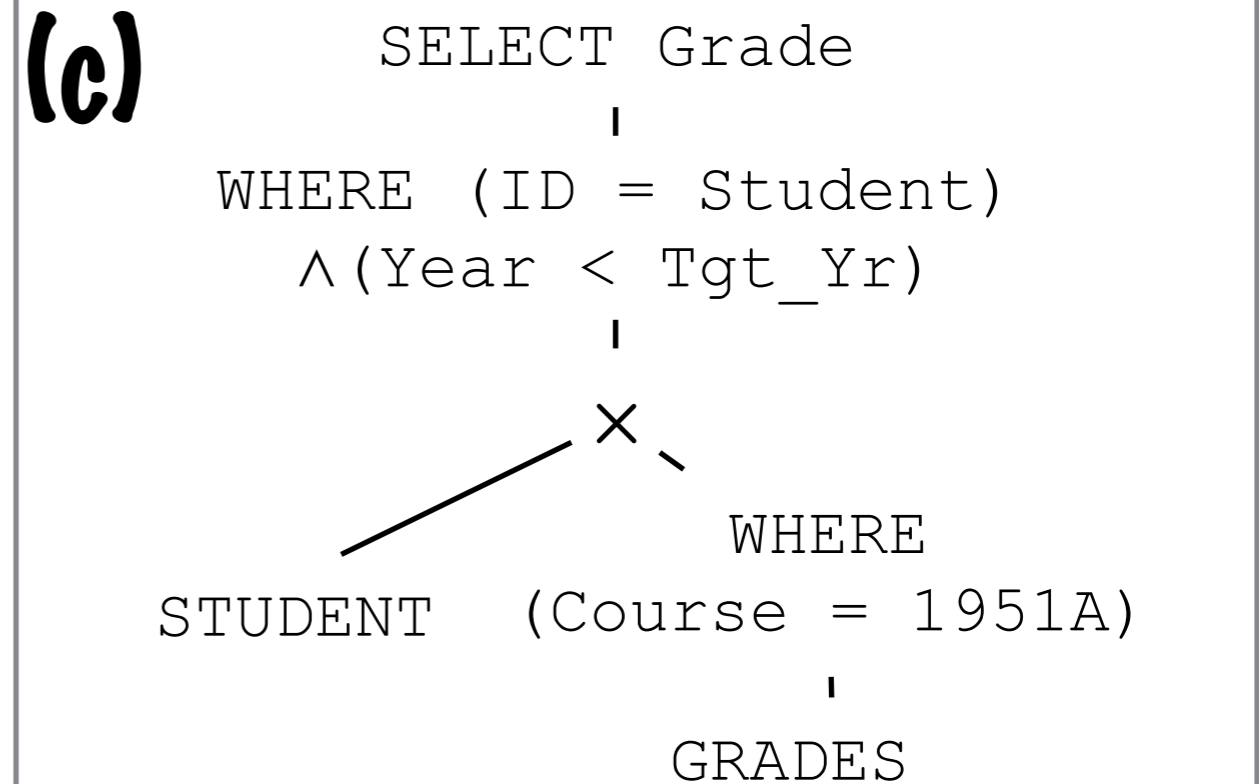
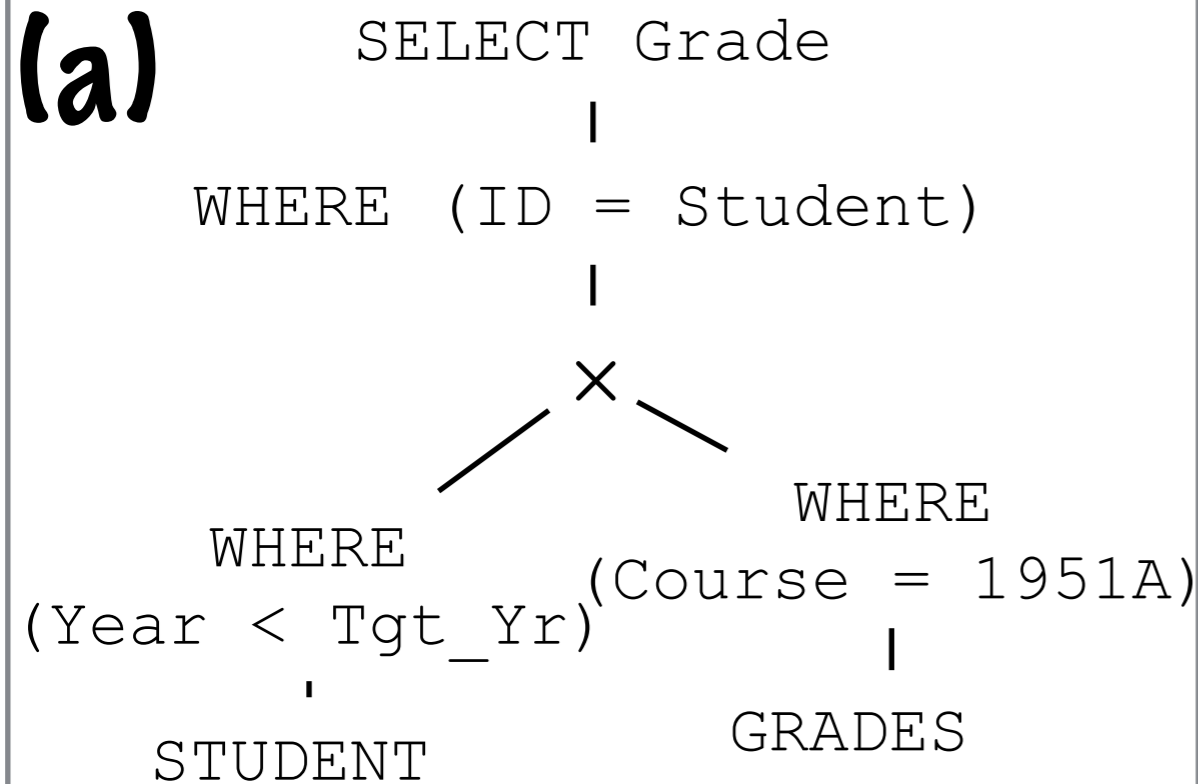
GRADES

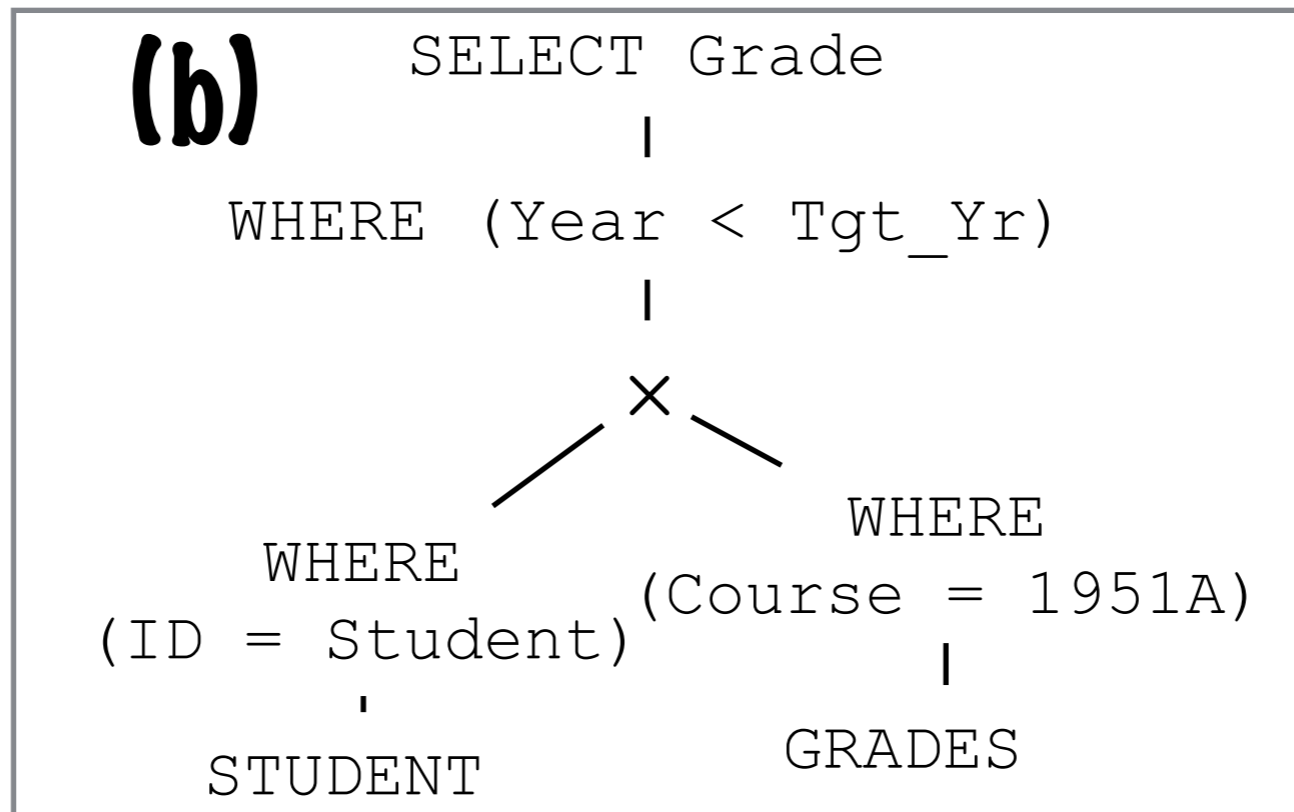
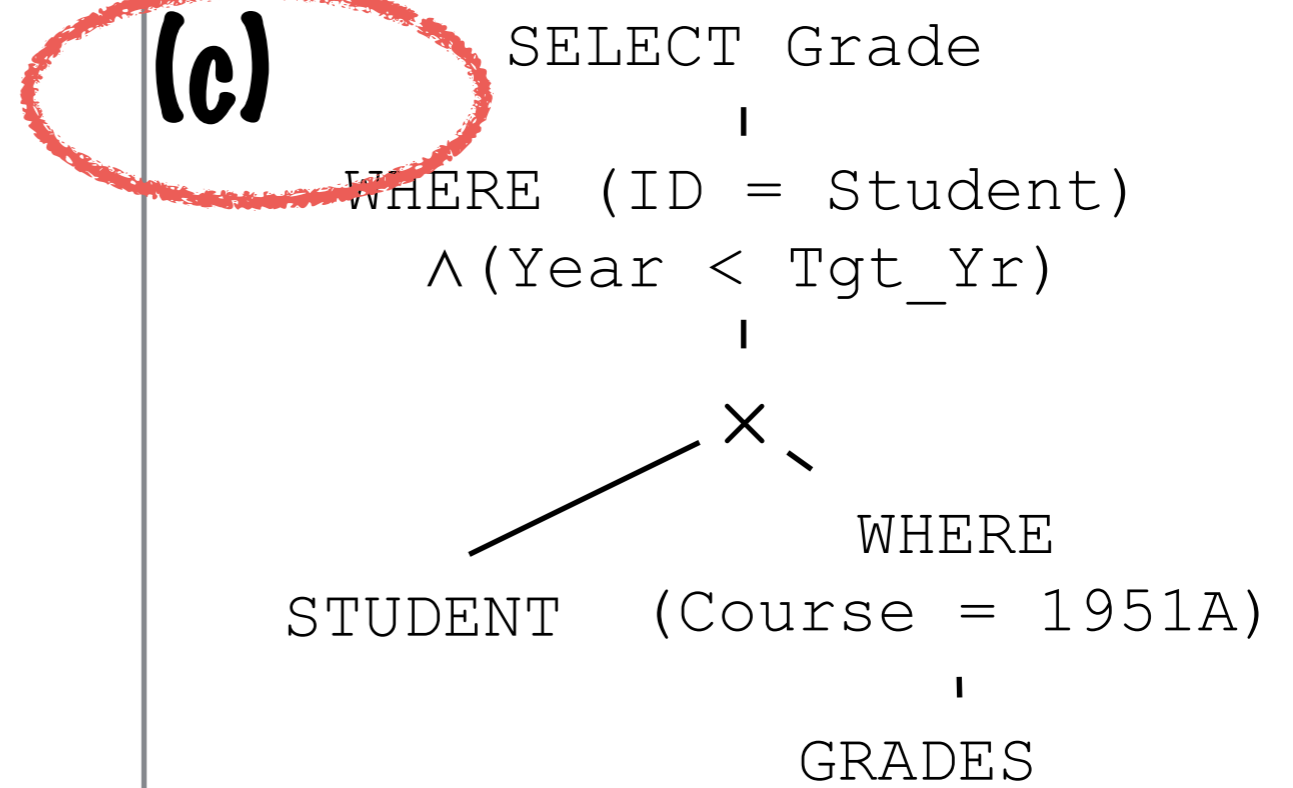
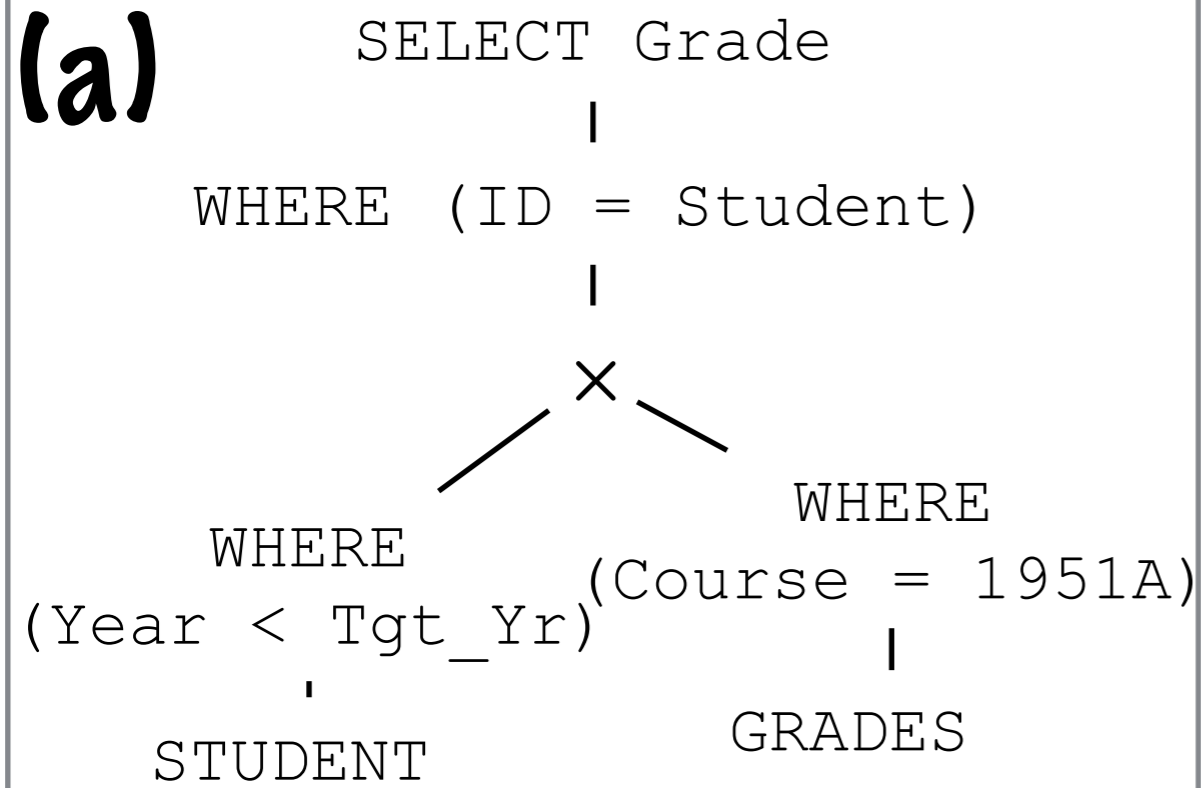
Student	Course	Grade	Tgt_Yr
1	32	A	1
2	1951A	A	3
6	32	A	1

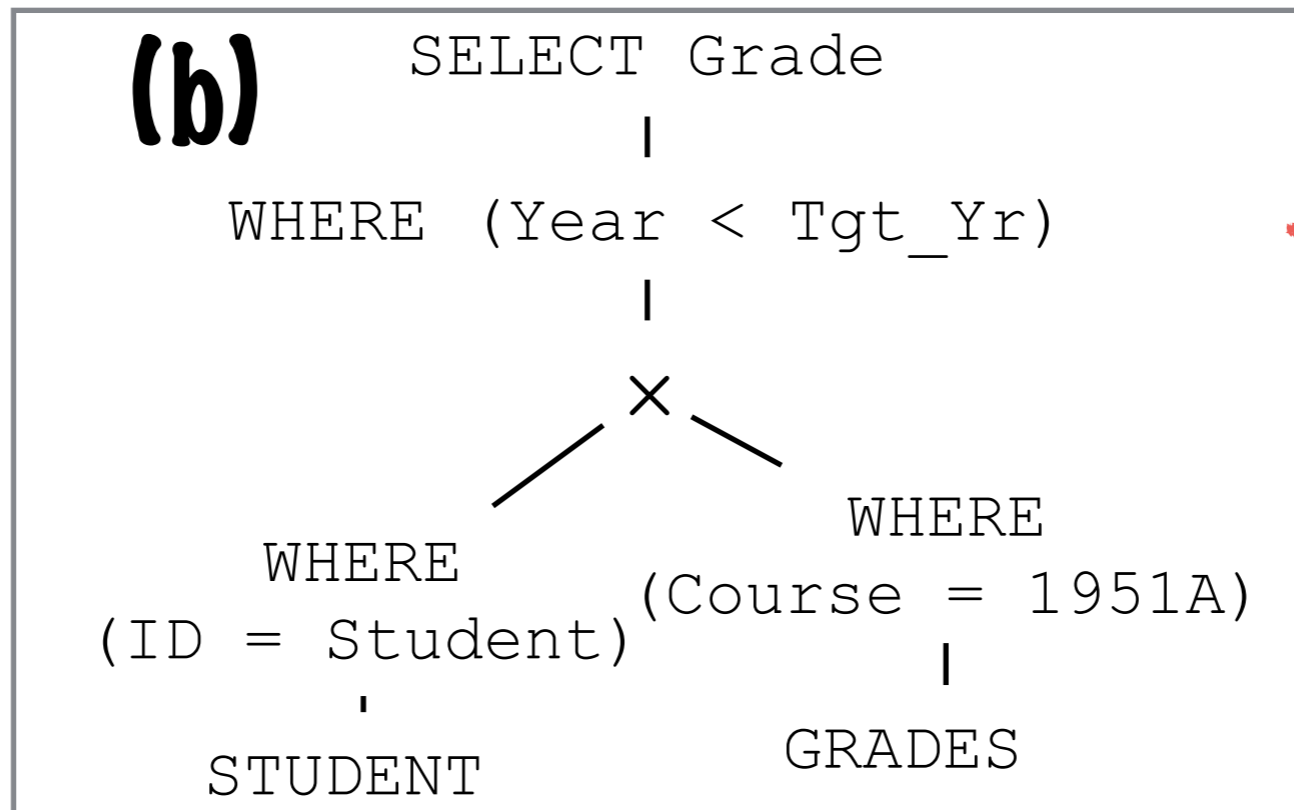
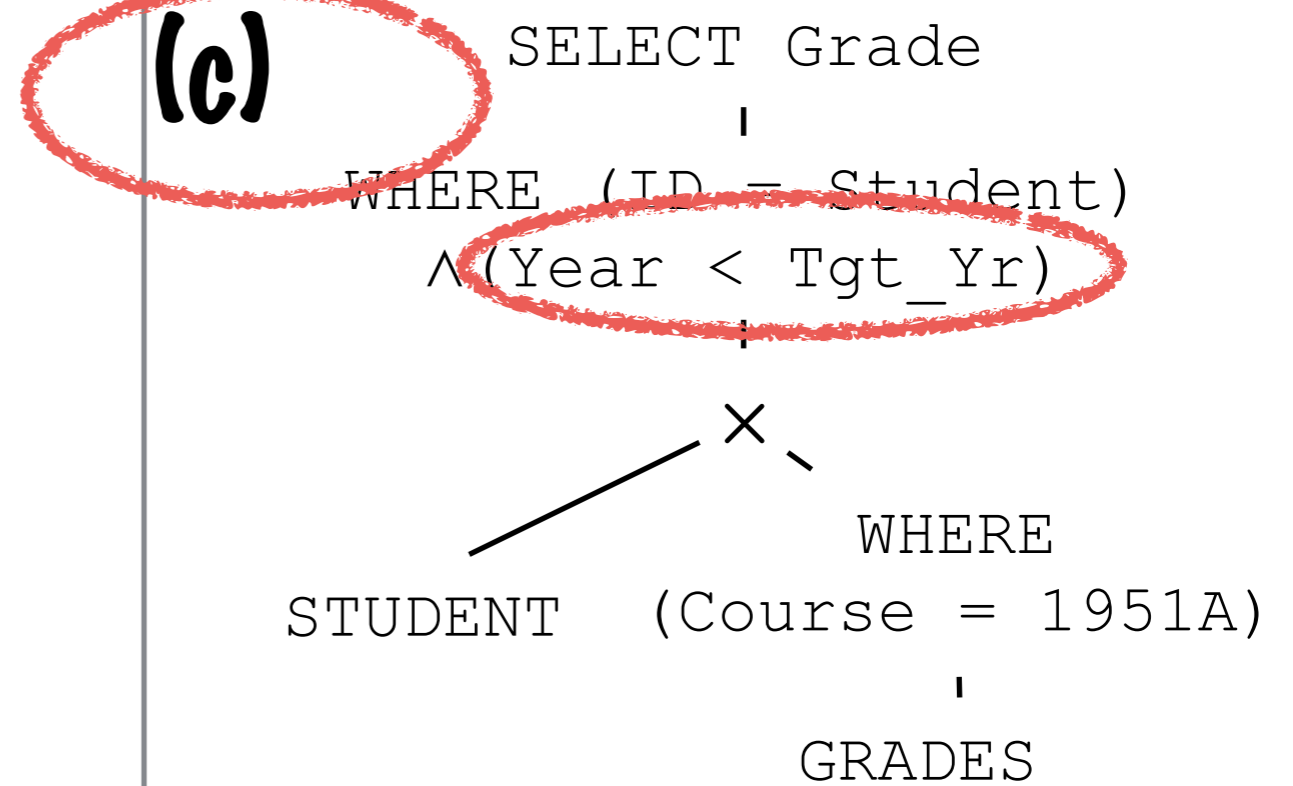
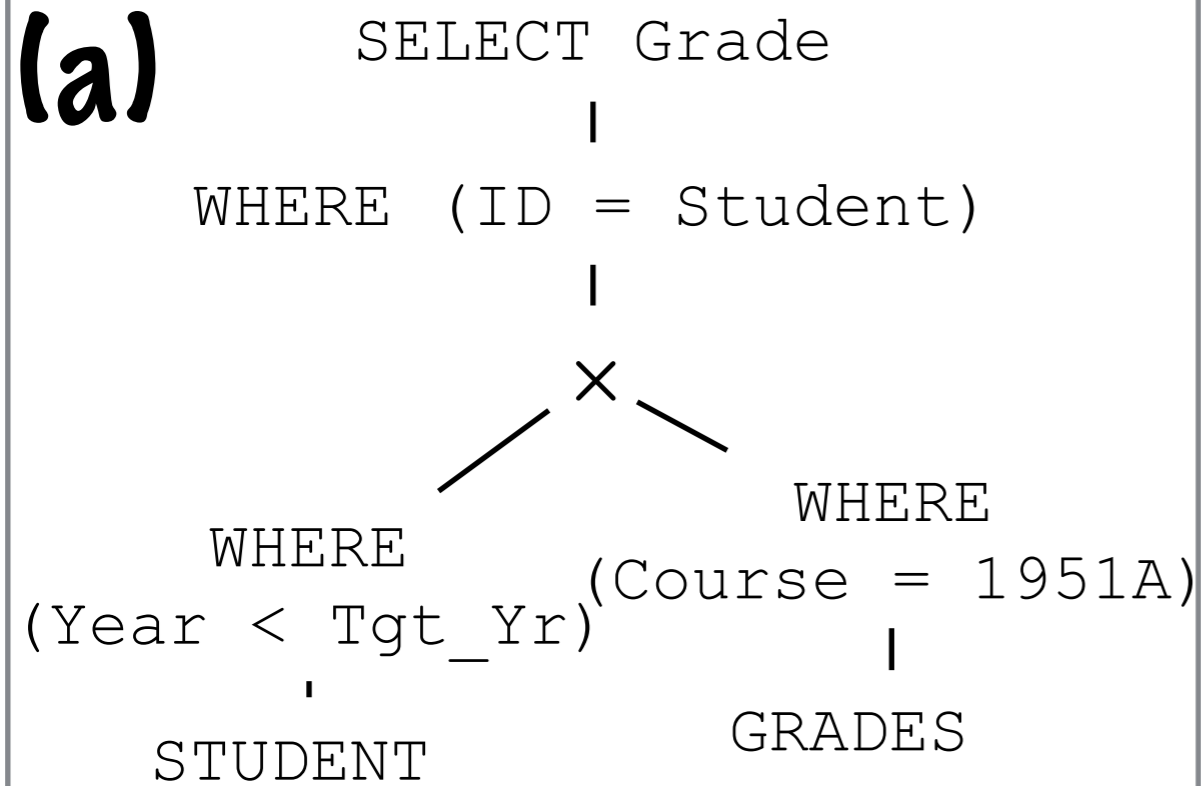
```
SELECT Grade
```

```
WHERE (ID = Student)
      ^ (Course = 1951A)
      ^ (Year < Tgt_Yr)
```









*Depends on output of join*

# Nested Queries

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

```
SELECT s.Name
FROM STUDENT s
WHERE NOT EXISTS (
  SELECT *
  FROM GRADES g
  WHERE s.ID = g.Student
)
```

Find names students who are not in any classes.

# Nested Queries

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

Outer  
Query

```
SELECT s.Name
FROM STUDENT s
WHERE NOT EXISTS (
  SELECT *
  FROM GRADES g
  WHERE s.ID = g.Student
)
```

Inner  
Query

Find names students who are not in any classes.

# Nested Queries

STUDENT


ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

Correlated!  
Inner query  
will return  
differently  
for every  
row...

```
SELECT s.Name
FROM STUDENT s
WHERE NOT EXISTS (
  SELECT *
  FROM GRADES g
  WHERE s.ID = g.Student
)
```



Find names students who are not in any classes.



# Nested Queries

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

*Not  
correlated!  
Inner query  
will always  
return the  
same thing.*

```
SELECT s.Name  
FROM STUDENT s  
WHERE s.ID NOT IN (  
    SELECT Student  
    FROM GRADES  
)
```

Find names students who are not in any  
classes.

# Clicker Question! (x2)

# Clicker Question!

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

How many courses is each student taking?

```
SELECT s.ID, s.Name,  
       (SELECT COUNT(*)  
        FROM GRADES g  
        WHERE s.ID = g.Student)  
FROM STUDENT s
```

**Is this query correlated?**

**(a) uh huh      (b) nuh uh**

# Clicker Question!

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

How many courses is each student taking?

```
SELECT s.ID, s.Name,  
       (SELECT COUNT(*)  
        FROM GRADES g  
        WHERE s.ID = g.Student)  
FROM STUDENT s
```

*Yes! This value will be  
different for every row  
(i.e. for every s.ID)*

**Is this query correlated?**

**(a) uh huh**      **(b) nuh uh**

# Clicker Question!

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

How many courses is each student taking?

```
SELECT s.ID, s.Name, c.num_courses
FROM STUDENT s,
  (SELECT Student,
   COUNT(*) AS num_courses
   FROM GRADES
   GROUP BY Student) c
WHERE s.ID = c.Student
```

**Is this query correlated?**

**(a) yeah sure    (b) not really**

# Clicker Question!

STUDENT

ID	Name	Year
1	Diane	4
2	Sol	5
3	Josh	5
4	Karlly	4
5	Mounik	4

GRADES

Studen	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

How many courses is each student taking?

```
SELECT s.ID, s.Name, c.num_courses
FROM STUDENT s,
  (SELECT Student,
   COUNT(*) AS num_courses
   FROM GRADES
   GROUP BY Student) c
WHERE s.ID = c.Student
```

*This value is always the same, regardless of the row*

Is this query correlated?

(a) yeah sure (b) not really

# **(non)Clicker Question!**

# (non)Clicker Question!

## Rewrite to remove the subquery altogether?

STUDENT

ID	Name	Year
1	Wennie	4
2	Maulik	5
3	Gurnaa	5
4	Jens	4
5	Erin	4

GRADES

Student	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

```
SELECT s.Name
FROM STUDENT s
WHERE EXISTS (
    SELECT * FROM GRADES
    WHERE s.ID = GRADES.Student
    AND s.Year < GRADES.Tgt_Yr
)
```

Find students taking courses that are above their level.



# (non)Clicker Question!

## Rewrite to remove the subquery altogether?

STUDENT

ID	Name	Year
1	Wennie	4
2	Maulik	5
3	Gurnaa	5
4	Jens	4
5	Erin	4

GRADES

Student	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

```
SELECT s.Name
FROM STUDENT s
WHERE EXISTS (
    SELECT * FROM GRADES
    WHERE s.ID = GRADES.Student
    AND s.Year < GRADES.Tgt_Yr
)
```

**HINT! Use a  
Join Condition**

Find students taking courses that are  
above their level.

# (non)Clicker Question!

## Rewrite to remove the subquery altogether?

STUDENT

ID	Name	Year
1	Wennie	4
2	Maulik	5
3	Gurnaa	5
4	Jens	4
5	Erin	4

GRADES

Student	Cours	GPA	Tgt_Yr
1	32	4.0	1
2	1951A	3.5	3
6	32	2.8	1

```
SELECT s.Name
FROM STUDENT s, GRADES g
WHERE s.ID = g.Student
      AND s.Year < g.Tgt_Yr
```

Find students taking courses that are  
above their level.

# Enough SQL...what about... NoSQL?

- NoSQL: no schema! just key-value stores
  - dictionaries instead of tables
  - (basically, jsons)
- Good for: fast development, flexibility, messy data
- Bad for: data integrity, safety guarantees
- “What about for my final project?” ....uh, ask me later

