

## Tables

---

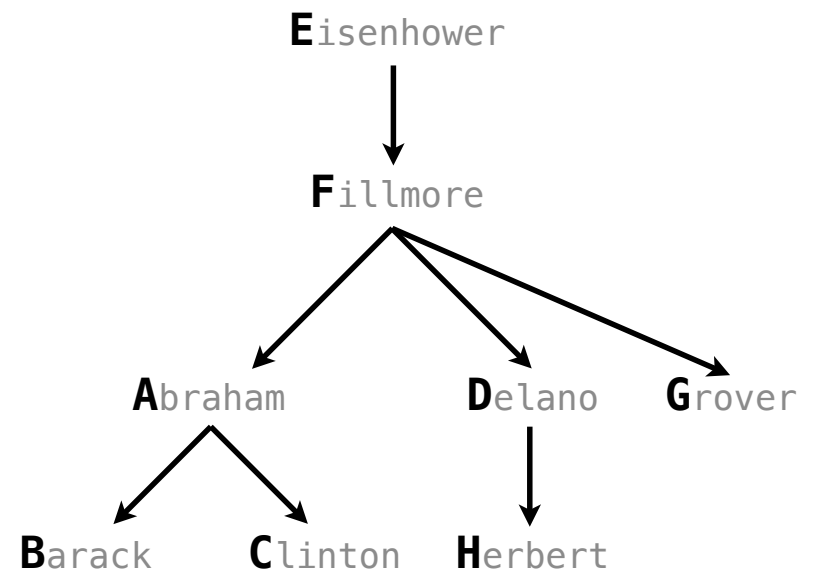
## Announcements



Recursive Art Contest  
Fall 2015

## Joining Tables

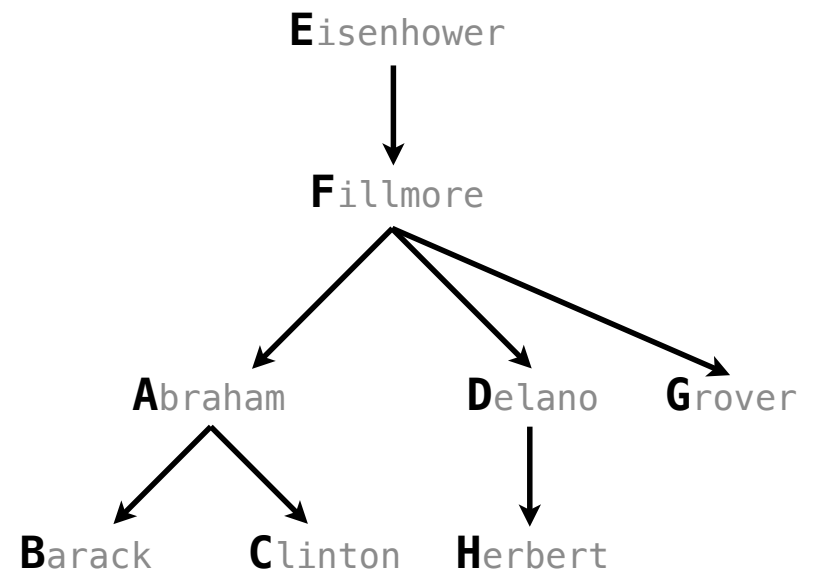
## Reminder: John the Patriotic Dog Breeder



## Reminder: John the Patriotic Dog Breeder



```
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham"      , "clinton"      UNION
SELECT "delano"        , "herbert"      UNION
SELECT "fillmore"      , "abraham"      UNION
SELECT "fillmore"      , "delano"       UNION
SELECT "fillmore"      , "grover"       UNION
SELECT "eisenhower"    , "fillmore";
```

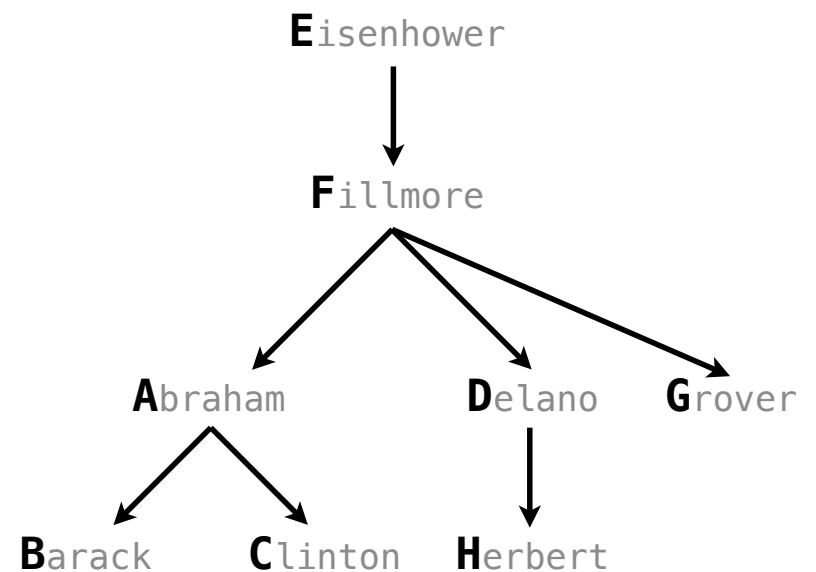


## Reminder: John the Patriotic Dog Breeder



CREATE TABLE parents AS

```
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham"      , "clinton"      UNION
SELECT "delano"        , "herbert"      UNION
SELECT "fillmore"      , "abraham"     UNION
SELECT "fillmore"      , "delano"      UNION
SELECT "fillmore"      , "grover"      UNION
SELECT "eisenhower"    , "fillmore";
```



## Reminder: John the Patriotic Dog Breeder



```
CREATE TABLE parents AS
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham"      , "clinton"      UNION
SELECT "delano"        , "herbert"     UNION
SELECT "fillmore"      , "abraham"   UNION
SELECT "fillmore"      , "delano"   UNION
SELECT "fillmore"      , "grover"   UNION
SELECT "eisenhower"    , "fillmore";
```

Parents :

Parent	Child
abraham	barack
abraham	clinton
delano	herbert
fillmore	abraham
fillmore	delano
fillmore	grover
eisenhower	fillmore



## Joining Two Tables

---

## Joining Two Tables

---

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

## Joining Two Tables

---

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS  
  SELECT "abraham" AS name, "long" AS fur UNION
```

**A**

## Joining Two Tables

---

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"   , "short"   UNION
```

**A**

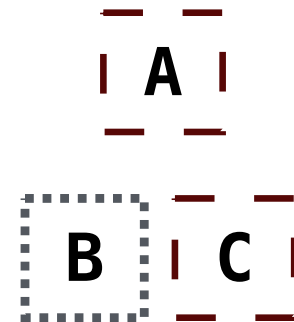
**B**

## Joining Two Tables

---

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"   , "short"      UNION
  SELECT "clinton"  , "long"       UNION
```

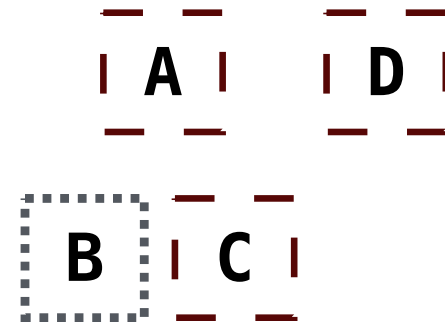


## Joining Two Tables

---

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"    , "short"      UNION
  SELECT "clinton"   , "long"       UNION
  SELECT "delano"     , "long"       UNION
```



## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
```



## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"     UNION
  SELECT "fillmore"    , "curly"      UNION
```

**E**

**F**

**A**   **D**

**B**   **C**



## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
```

**E**

**F**

**A**

**D**

**G**

**B**

**C**

## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"     UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"     UNION
  SELECT "herbert"     , "curly";
```

**E**

**F**

**A**

**D**

**G**

**B**

**C**

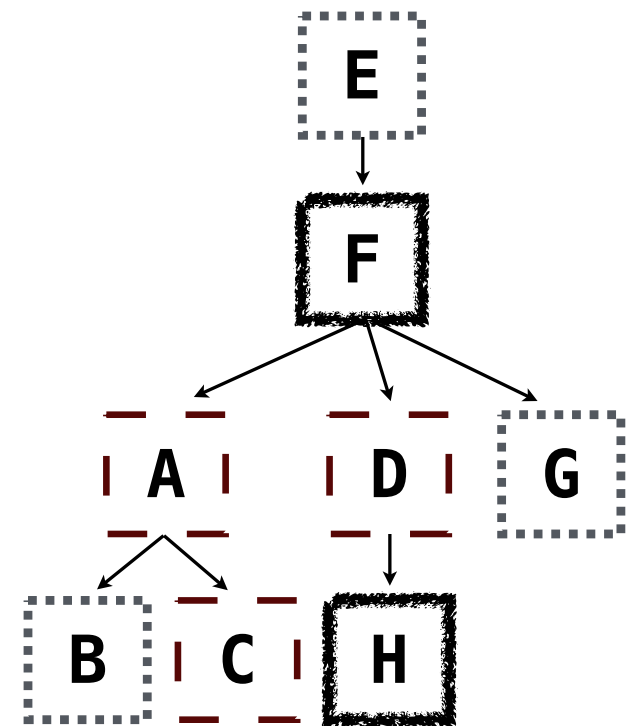
**H**

## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
  SELECT "herbert"     , "curly";

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"  UNION
  ...;
```



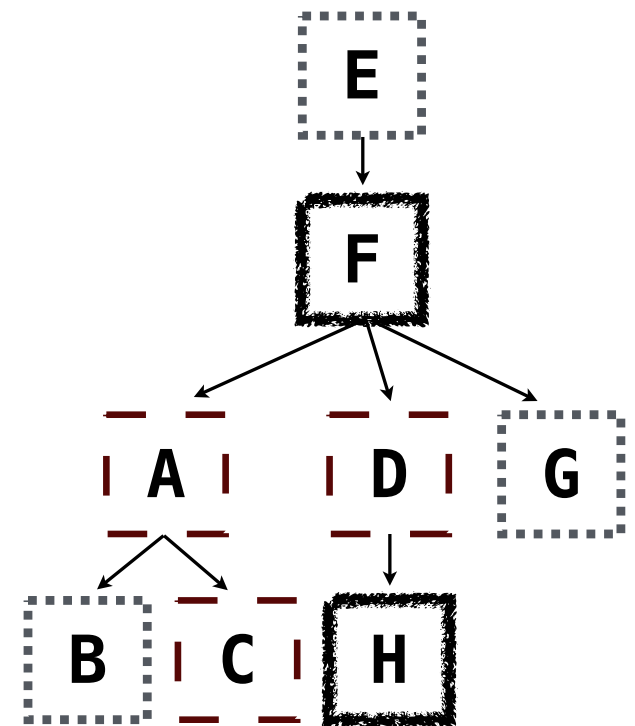
## Joining Two Tables

Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
  SELECT "herbert"     , "curly";

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"  UNION
  ...;
```

Select the parents of curly-furred dogs



## Joining Two Tables

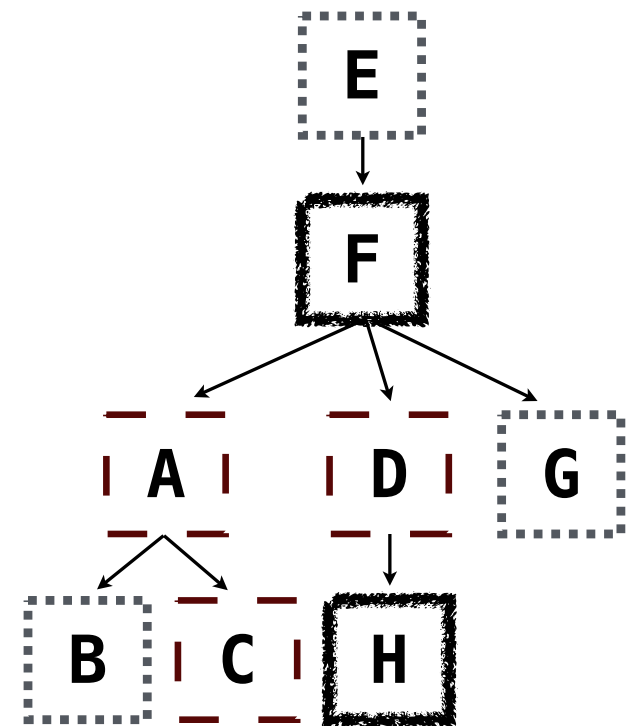
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"      , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
  SELECT "herbert"     , "curly";

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"  UNION
  ...;
```

Select the parents of curly-furred dogs

```
SELECT parent FROM parents, dogs
      WHERE child = name AND fur = "curly";
```



## Joining Two Tables

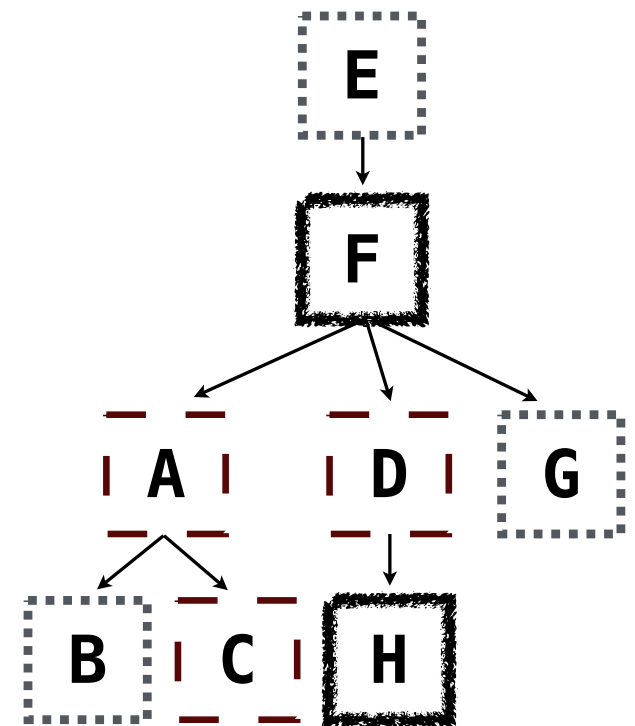
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
  SELECT "herbert"     , "curly";

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"  UNION
  ...;
```

Select the parents of curly-furred dogs

```
SELECT parent FROM parents, dogs
WHERE child = name AND fur = "curly";
```



## Joining Two Tables

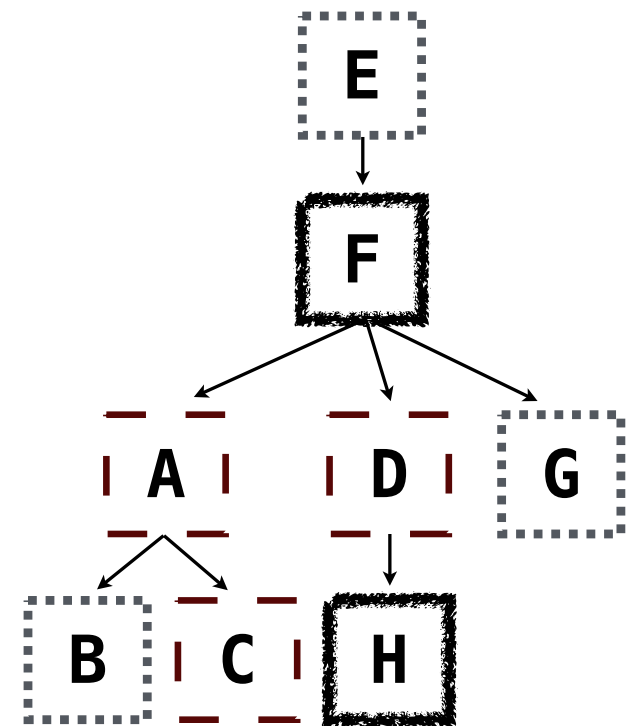
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  SELECT "clinton"     , "long"       UNION
  SELECT "delano"       , "long"       UNION
  SELECT "eisenhower"  , "short"      UNION
  SELECT "fillmore"    , "curly"      UNION
  SELECT "grover"       , "short"      UNION
  SELECT "herbert"     , "curly";

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"  UNION
  ...;
```

Select the parents of curly-furred dogs

```
SELECT parent FROM parents, dogs
WHERE child = name AND fur = "curly";
```

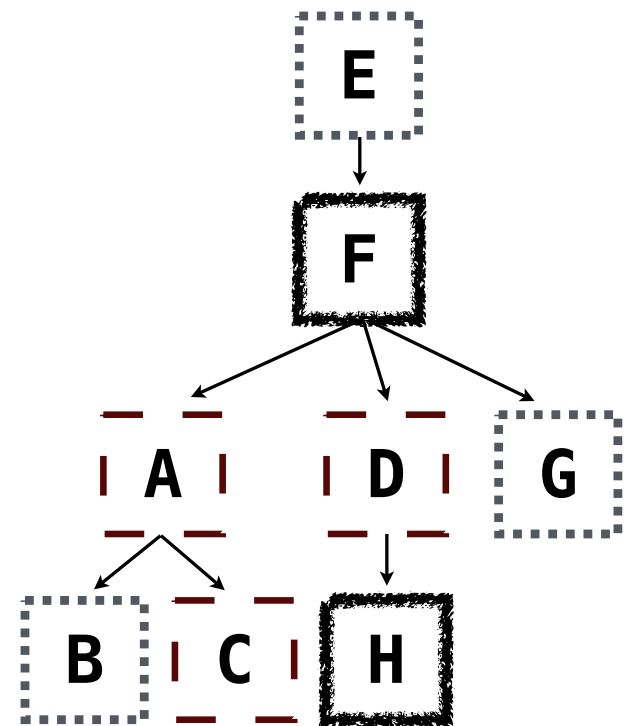


## Aliases and Dot Expressions



## Joining a Table with Itself

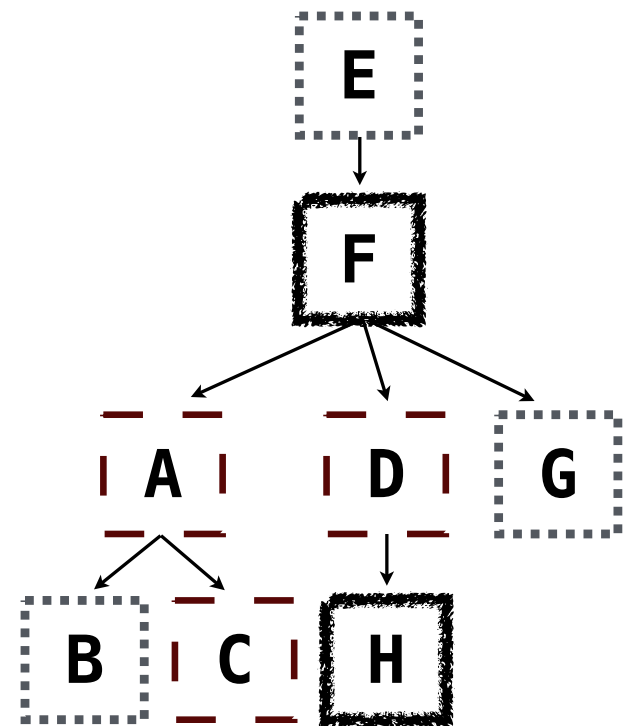
---



## Joining a Table with Itself

---

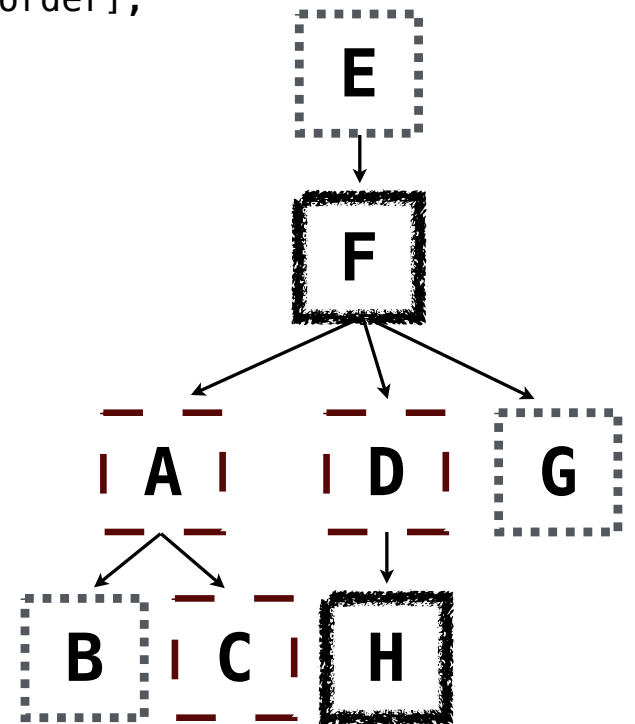
Two tables may share a column name; dot expressions and aliases disambiguate column values



## Joining a Table with Itself

Two tables may share a column name; dot expressions and aliases disambiguate column values

```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

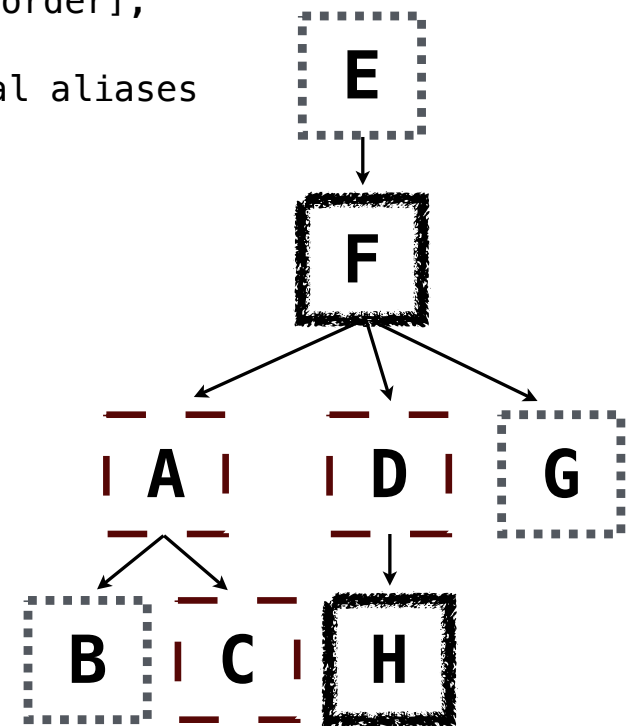


## Joining a Table with Itself

Two tables may share a column name; dot expressions and aliases disambiguate column values

```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

[table] is a comma-separated list of table names with optional aliases



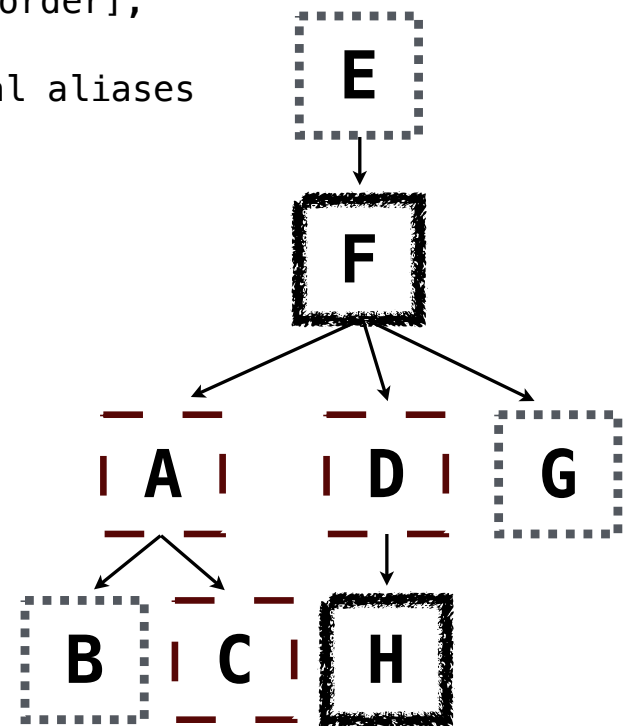
## Joining a Table with Itself

Two tables may share a column name; dot expressions and aliases disambiguate column values

```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

[table] is a comma-separated list of table names with optional aliases

Select all pairs of siblings



## Joining a Table with Itself

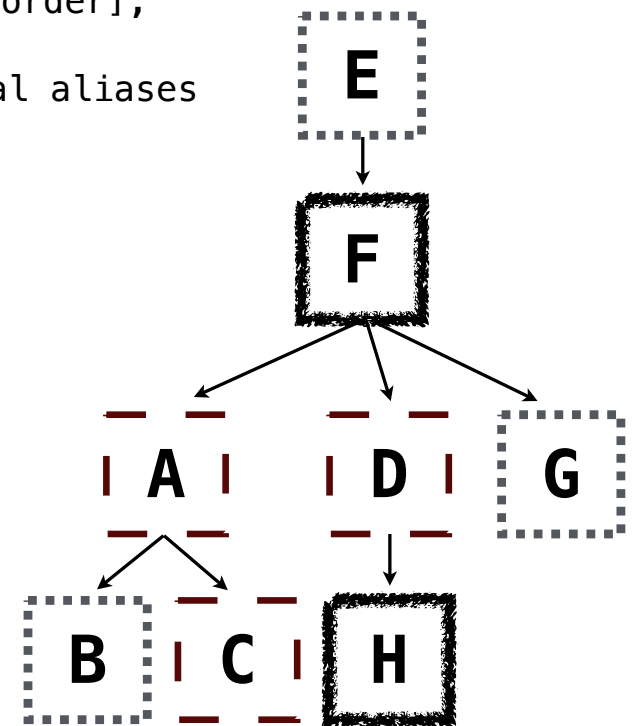
Two tables may share a column name; dot expressions and aliases disambiguate column values

```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

[table] is a comma-separated list of table names with optional aliases

Select all pairs of siblings

```
SELECT a.child AS first, b.child AS second  
FROM parents AS a, parents AS b  
WHERE a.parent = b.parent AND a.child < b.child;
```



## Joining a Table with Itself

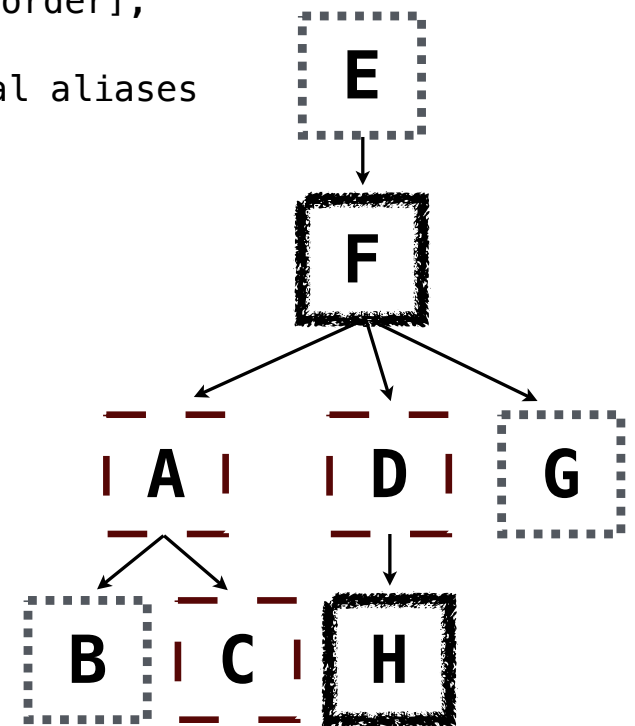
Two tables may share a column name; dot expressions and aliases disambiguate column values

```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

[table] is a comma-separated list of table names with optional aliases

Select all pairs of siblings

```
SELECT a.child AS first, b.child AS second  
FROM parents AS a, parents AS b  
WHERE a.parent = b.parent AND a.child < b.child;
```



## Joining a Table with Itself

Two tables may share a column name; dot expressions and aliases disambiguate column values

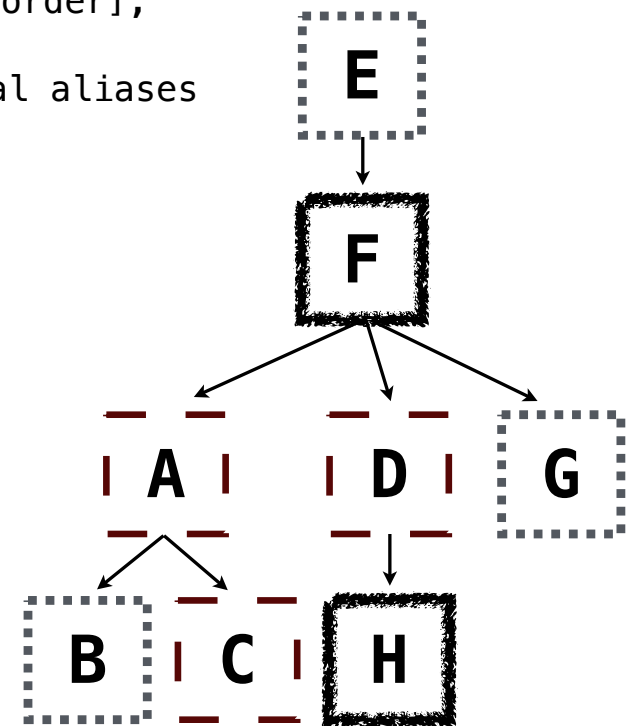
```
SELECT [columns] FROM [table] WHERE [condition] ORDER BY [order];
```

[table] is a comma-separated list of table names with optional aliases

Select all pairs of siblings

```
SELECT a.child AS first, b.child AS second  
FROM parents AS a, parents AS b  
WHERE a.parent = b.parent AND a.child < b.child;
```

first	second
barack	clinton
abraham	delano
abraham	grover
delano	grover

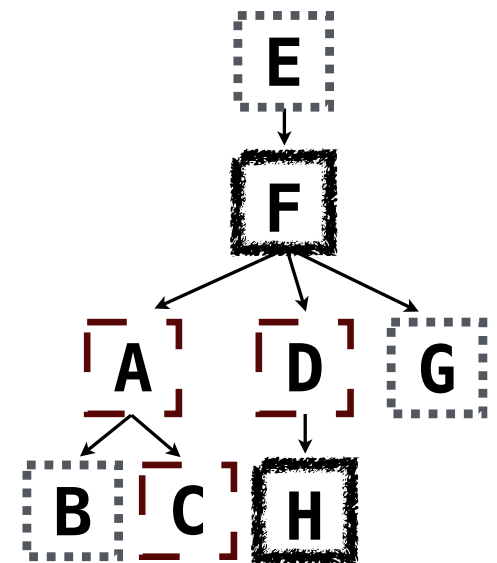




## Example: Grandparents

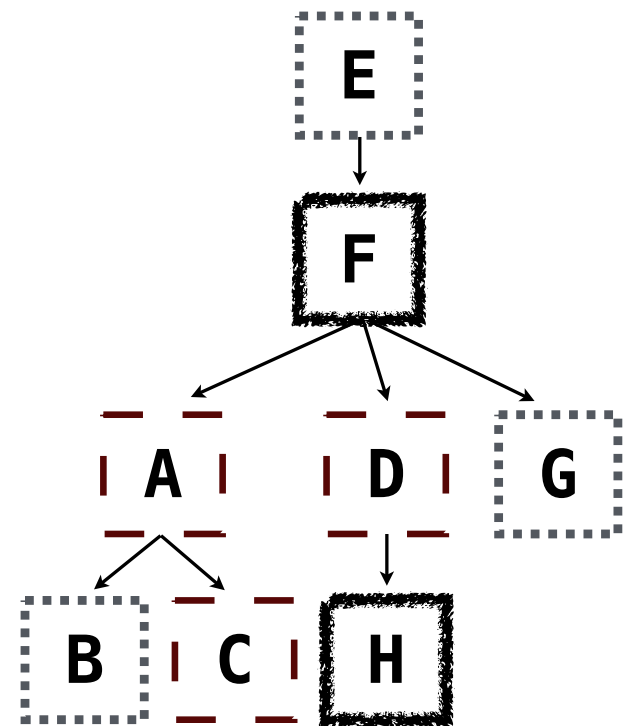
Which select statement evaluates to all grandparent, grandchild pairs?

- 1 `SELECT a.grandparent, b.child FROM parents AS a, parents AS b  
WHERE b.parent = a.child;`
- 2 `SELECT a.parent, b.child FROM parents AS a, parents AS b  
WHERE a.parent = b.child;`
- 3 `SELECT a.parent, b.child FROM parents AS a, parents AS b  
WHERE b.parent = a.child;`
- 4 `SELECT a.grandparent, b.child FROM parents AS a, parents AS b  
WHERE a.parent = b.child;`
- 5 None of the above



## Joining Multiple Tables

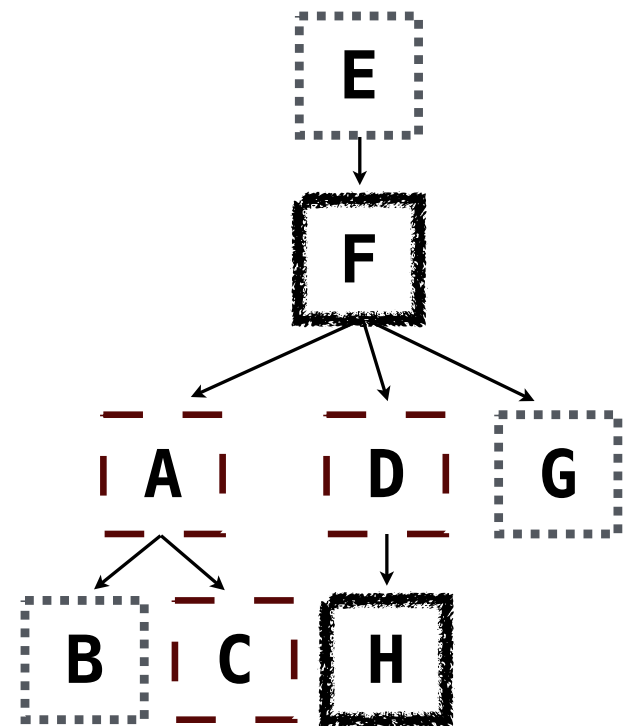
---



## Joining Multiple Tables

---

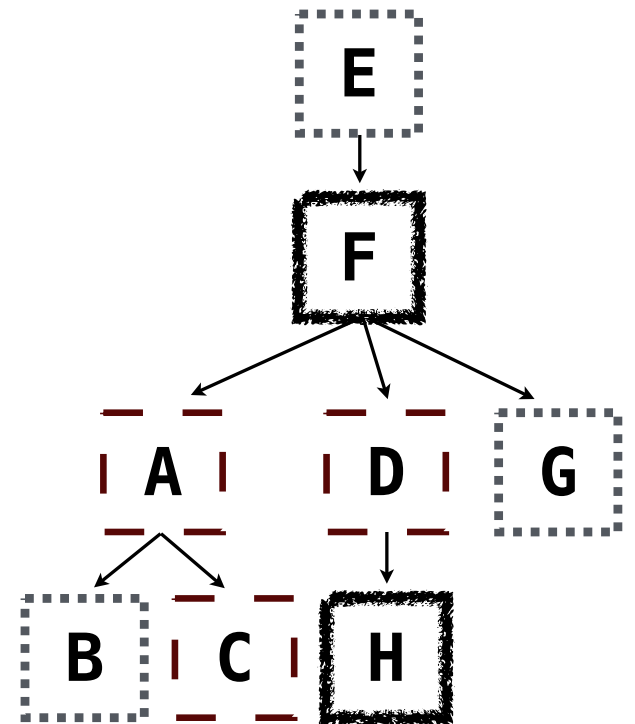
Multiple tables can be joined to yield all combinations of rows from each



## Joining Multiple Tables

Multiple tables can be joined to yield all combinations of rows from each

```
CREATE TABLE grandparents AS
SELECT a.parent AS granddog, b.child AS granpup
FROM parents AS a, parents AS b
WHERE b.parent = a.child;
```

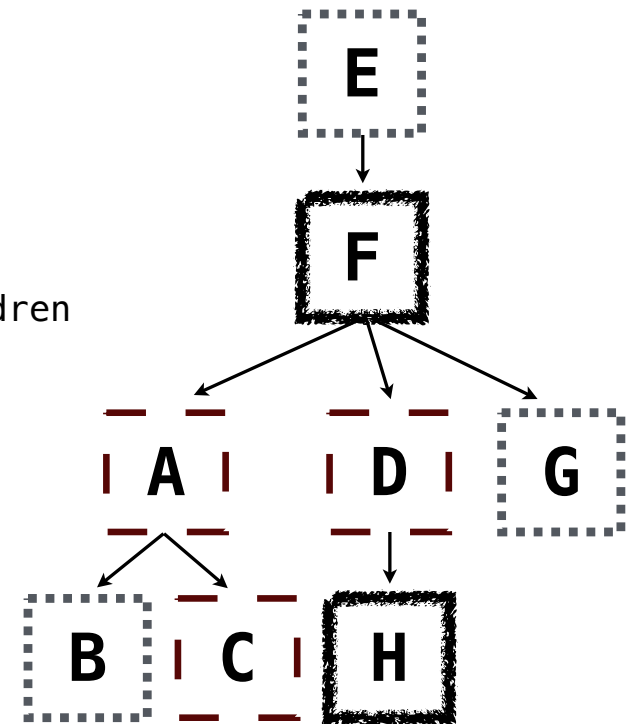


## Joining Multiple Tables

Multiple tables can be joined to yield all combinations of rows from each

```
CREATE TABLE grandparents AS
SELECT a.parent AS granddog, b.child AS granpup
FROM parents AS a, parents AS b
WHERE b.parent = a.child;
```

Select all grandparents with the same fur as their grandchildren



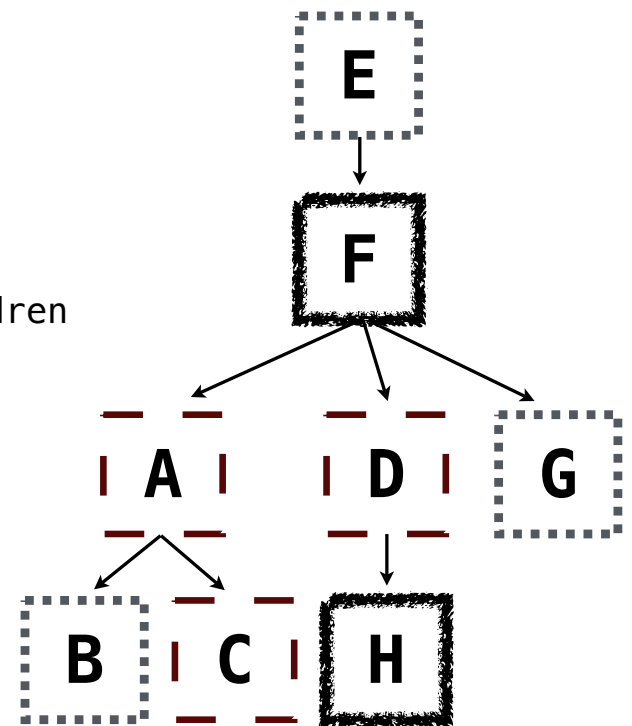
## Joining Multiple Tables

Multiple tables can be joined to yield all combinations of rows from each

```
CREATE TABLE grandparents AS
SELECT a.parent AS granddog, b.child AS granpup
FROM parents AS a, parents AS b
WHERE b.parent = a.child;
```

Select all grandparents with the same fur as their grandchildren

Which tables need to be joined together?



## Joining Multiple Tables

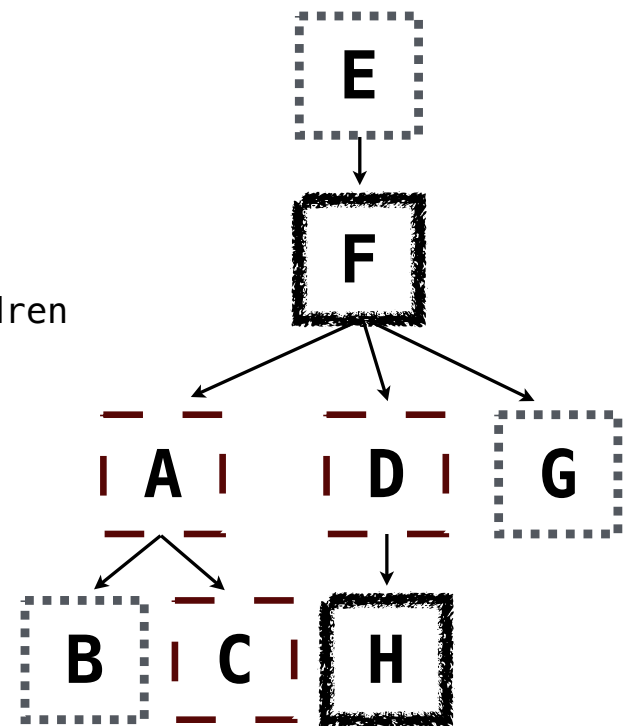
Multiple tables can be joined to yield all combinations of rows from each

```
CREATE TABLE grandparents AS
SELECT a.parent AS grandog, b.child AS granpup
FROM parents AS a, parents AS b
WHERE b.parent = a.child;
```

Select all grandparents with the same fur as their grandchildren

Which tables need to be joined together?

```
SELECT grandog FROM grandparents, dogs AS c, dogs AS d
WHERE grandog = c.name AND
      granpup = d.name AND
      c.fur = d.fur;
```



Example: Dog Triples



## Fall 2014 Quiz Question (Slightly Modified)

---

## Fall 2014 Quiz Question (Slightly Modified)

---

Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

## Fall 2014 Quiz Question (Slightly Modified)

---

Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  ...;
```

## Fall 2014 Quiz Question (Slightly Modified)

---

Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  ...;

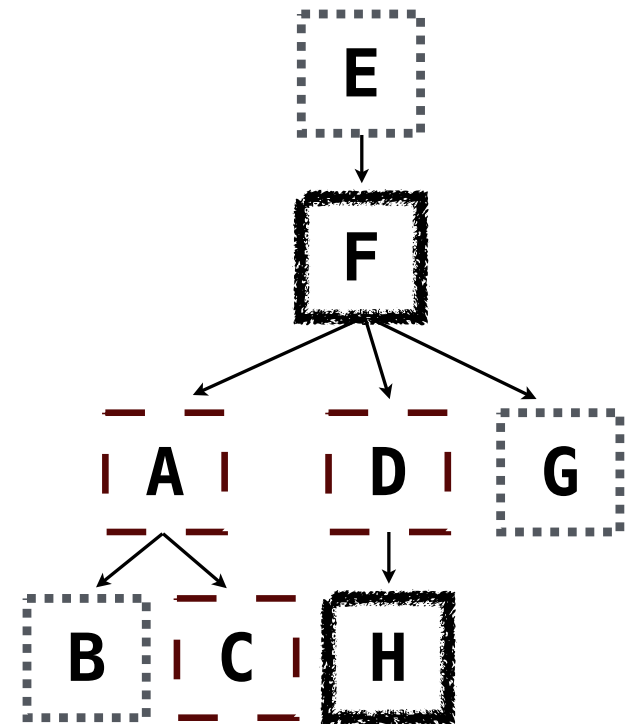
CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"      UNION
  ...;
```

## Fall 2014 Quiz Question (Slightly Modified)

Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  ...;

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"      UNION
  ...;
```



## Fall 2014 Quiz Question (Slightly Modified)

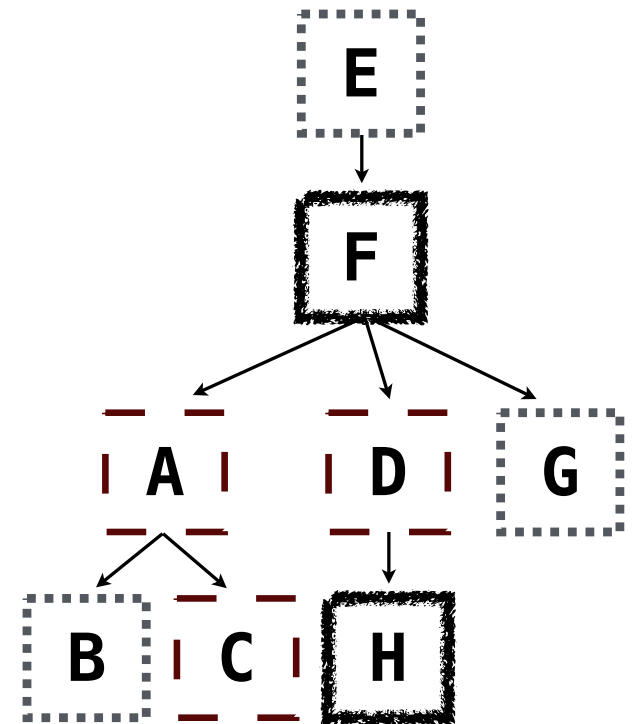
Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  ...;

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"      UNION
  ...;
```

Expected output:

```
delano|clinton|abraham
grover|eisenhower|barack
```



## Fall 2014 Quiz Question (Slightly Modified)

Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

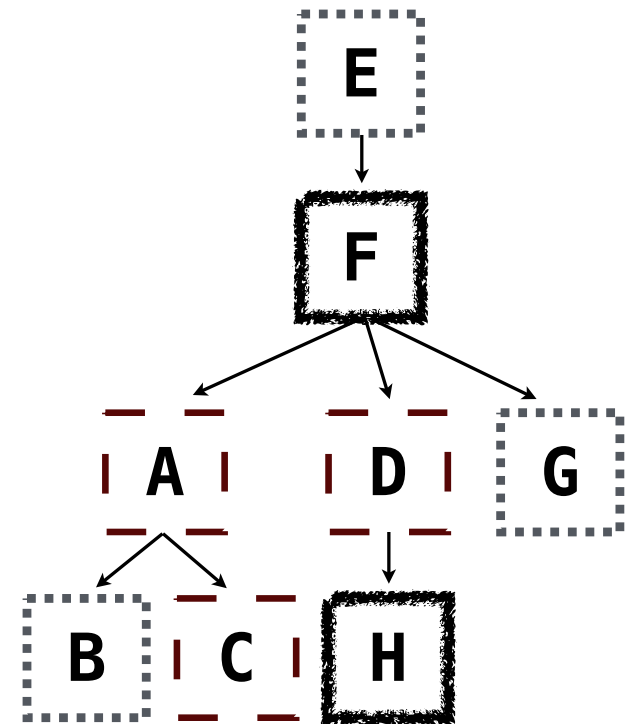
```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack"      , "short"      UNION
  ...;

CREATE TABLE parents AS
  SELECT "abraham" AS parent, "barack" AS child UNION
  SELECT "abraham"      , "clinton"      UNION
  ...;
```

Expected output:

```
delano|clinton|abraham
grover|eisenhower|barack
```

(Demo)



## Numerical Expressions



## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
[expression] AS [name], [expression] AS [name], ...
```

```
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
[expression] AS [name], [expression] AS [name], ...  
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

Combine values: +, -, \*, /, %, and, or

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
[expression] AS [name], [expression] AS [name], ...  
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

Combine values: +, -, \*, /, %, and, or

Transform values: abs, round, not, -

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
[expression] AS [name], [expression] AS [name], ...
```

```
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

Combine values: +, -, \*, /, %, and, or

Transform values: abs, round, not, -

Compare values: <, <=, >, >=, <>, !=, =

## Numerical Expressions

---

Expressions can contain function calls and arithmetic operators

```
[expression] AS [name], [expression] AS [name], ...
```

```
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

Combine values: +, -, \*, /, %, and, or

Transform values: abs, round, not, -

Compare values: <, <=, >, >=, <>, !=, =

(Demo)

## String Expressions



## String Expressions

---

String values can be combined to form longer strings

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python

```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python

```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python

```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea



## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea

```
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea

```
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;  
sqlite> SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cadr FROM lists;
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea

```
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;  
sqlite> SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cadr FROM lists;  
two
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea



```
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;  
sqlite> SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cadr FROM lists;  
two
```

## String Expressions

---

String values can be combined to form longer strings



```
sqlite> SELECT "hello," || " world";  
hello, world
```

Basic string manipulation is built into SQL, but differs from Python



```
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;  
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;  
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea



```
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;  
sqlite> SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cadr FROM lists;  
two
```

(Demo)