CS186 Discussion #2

(SQL)

SELECT [DISTINCT] <column list>
FROM <relation list>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]]
[**ORDER BY** <column list>];

```
SELECT name
FROM Students;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

```sql
SELECT name
FROM Students;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

name
Bob
Sue
Ron
Al
Sally
Bob

SELECT DISTINCT name **FROM** Students;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

```
SELECT DISTINCT name
FROM Students;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

$\Rightarrow$

name
Bob
Sue
Ron
Al
Sally

```
SELECT DISTINCT name
FROM Students
WHERE gpa > 3.0;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

SELECT DISTINCT name
FROM Students
WHERE gpa > 3.0;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

→

name
Bob
Al
Sally

```
SELECT name, gpa
FROM Students
WHERE gpa > 3.0
ORDER BY gpa DESC;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

SELECT name, gpa
FROM Students
WHERE gpa > 3.0
ORDER BY gpa DESC;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

→

name	gpa
Al	4.0
Bob	3.7
Sally	3.6

```
SELECT name, gpa
FROM Students
WHERE gpa > 3.0
ORDER BY gpa DESC;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

name	gpa
Al	4.0

SELECT name, gpa
FROM Students
WHERE gpa > 3.0
ORDER BY gpa DESC LIMIT 1;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

name	gpa
Al	4.0

```
SELECT AVG(gpa)
FROM Students;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

```
SELECT AVG(gpa)
FROM Students;
```

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

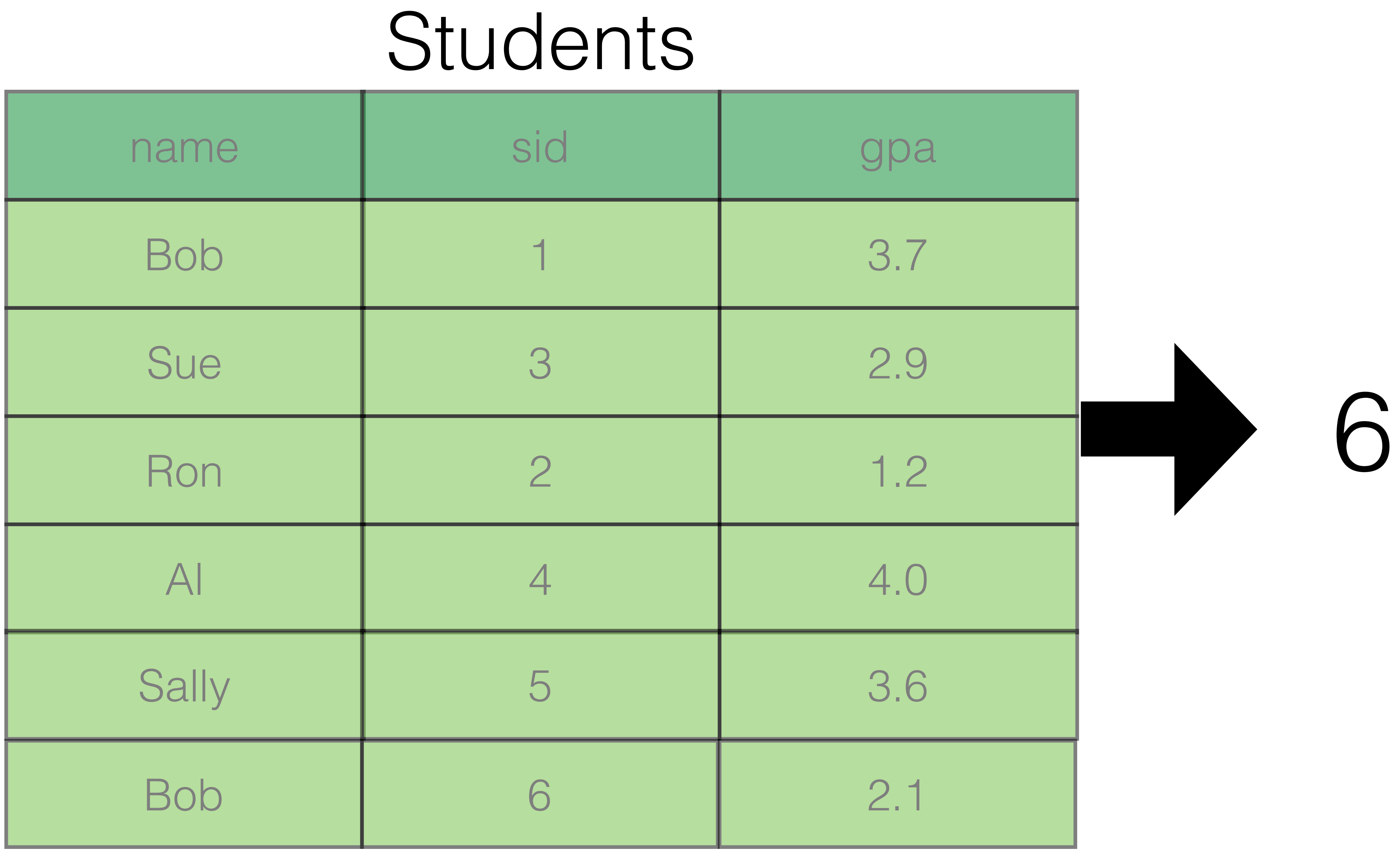
$\Rightarrow$ 2.9166..

SELECT COUNT(name)
FROM Students;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

SELECT COUNT(name) **FROM** Students;


Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

SELECT COUNT(DISTINCT name)
FROM Students;

Students

name	sid	gpa
Bob	1	3.7
Sue	3	2.9
Ron	2	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

→ 5

SELECT AVG(gpa)
FROM Students
GROUP BY dept;

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

SELECT AVG(gpa)
FROM Students
GROUP BY dept;

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

→

AVG(gpa)
3.7
3.45
2.4
2.1

```
SELECT dept, AVG(gpa)
FROM Students
GROUP BY dept;
```

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

→

dept	AVG(gpa)
english	3.7
eecs	3.45
math	2.4
physics	2.1

```
SELECT name
FROM Students
GROUP BY dept;
```

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

SELECT name
FROM Students
GROUP BY dept;

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

→

name
Bob
Al
Sally
Bob

SELECT dept, AVG(gpa)
FROM Students
GROUP BY dept
HAVING AVG(gpa) > 3.0;

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

```
SELECT dept, AVG(gpa)
FROM Students
GROUP BY dept
HAVING AVG(gpa) > 3.0;
```

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

$\Rightarrow$

dept	AVG(gpa)
english	3.7
eecs	3.45

SELECT s.name
FROM Students s
WHERE s.dept **IN**
('eecs', 'math')**;**

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

$\rightarrow$

dept
Sue
Ron
Al
Sally

SELECT s.name
FROM Students s
WHERE s.dept **IN**
(**SELECT** dept **FROM** Students **WHERE** gpa > 3.0)**;**

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

```sql
SELECT s.name
FROM Students s
WHERE s.dept IN
(SELECT dept FROM Students WHERE gpa > 3.0);
```

Students

name	dept	gpa
Bob	english	3.7
Sue	eecs	2.9
Ron	math	1.2
Al	eecs	4.0
Sally	math	3.6
Bob	physics	2.1

dept
Bob
Sue
Ron
Al
Sally

Worksheet Part A

SELECT [DISTINCT] <column list>
FROM <relation list>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]]
[**ORDER BY** <column list>];

Find the 5 songs that spent the most weeks in the top 40, ordered from most to least.

Find the 5 songs that spent the most weeks in the top 40, ordered from most to least.

```
SELECT song_name FROM Songs ORDER
BY weeks_in_top_40 DESC LIMIT 5;
```

Find the name and first year active of every artist whose name starts with the letter ‘B’.

Find the name and first year active of every artist whose name starts with the letter ‘B’.

```
SELECT artist_name,
first_year_active FROM Artists
WHERE artist_name LIKE ‘B%’;
```

Find the total number of “Techno” albums released each year.

Find the total number of “Techno” albums released each year.

```
SELECT year_released, COUNT(*) FROM
Albums WHERE genre = ‘Techno’ GROUP
BY year_released;
```

Find the genre and the number of albums released per genre; don’t include genres that have a count of less than 10.

Find the genre and the number of albums released per genre; don’t include genres that have a count of less than 10.

```
SELECT genre, COUNT(*) FROM Albums
GROUP BY genre HAVING COUNT(*) >=
10;
```

Students

name	id	gpa
Bob	1	3.7
Sue	2	2.9
Ron	3	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

Grades

id	class	grade
1	cs186	C
1	cs164	A
3	cs70	B
4	cs61a	A
2	cs61c	D
4	cs170	A

SELECT S.name
FROM Students S, Grades G;

Students

name	id	gpa
Bob	1	3.7
Sue	2	2.9
Ron	3	1.2
Al	4	4.0
Sally	5	3.6
Bob	6	2.1

Grades

id	class	grade
1	cs186	C
1	cs164	A
3	cs70	B
4	cs61a	A
2	cs61c	D
4	cs170	A

```
SELECT S.name
FROM Students S, Grades G;
```

S.name	S.id	S.gpa	G.id	G.class	G.grade
Bob	1	3.7	1	cs186	C
Bob	1	3.7	1	cs164	A
Bob	1	3.7	3	cs70	B
Bob	1	3.7	4	cs61a	A
Bob	1	3.7	2	cs61c	D
Bob	1	3.7	4	cs170	A
Sue	2	2.9	1	cs186	C
Sue	2	2.9	1	cs164	A
Sue	2	2.9	3	cs70	B
Sue	2	2.9	4	cs61a	A
Sue	2	2.9	2	cs61c	D
Sue	2	2.9	4	cs170	A
Ron	3	1.2	1	cs186	C
Ron	3	1.2	1	cs164	A
Ron	3	1.2	3	cs70	B

```
SELECT S.name
FROM Students S, Grades G;
```

S.name	S.id	S.gpa	G.id	G.class	G.grade
Bob	1	3.7	1	cs186	C
Bob	1	3.7	1	cs164	A
Bob	1	3.7	3	cs70	B
Bob	1	3.7	4	cs61a	A
Bob	1	3.7	2	cs61c	D
Bob	1	3.7	4	cs170	A
Sue	2	2.9	1	cs186	C
Sue	2	2.9	1	cs164	A
Sue	2	2.9	3	cs70	B
Sue	2	2.9	4	cs61a	A
Sue	2	2.9	2	cs61c	D
Sue	2	2.9	4	cs170	A
Ron	3	1.2	1	cs186	C
Ron	3	1.2	1	cs164	A
Ron	3	1.2	3	cs70	B

SELECT S.name
FROM Students S, Grades G
WHERE S.id = G.id;

S.name	S.id	S.gpa	G.id	G.class	G.grade
Bob	1	3.7	1	cs186	C
Bob	1	3.7	1	cs164	A
Sue	2	2.9	2	cs61c	D
Ron	3	1.2	3	cs70	B
Al	4	4.0	4	cs61a	A
Al	4	4.0	4	cs170	A

SELECT S.name
FROM Students S, Grades G
WHERE S.id = G.id AND G.grade = ‘A’;

S.name	S.id	S.gpa	G.id	G.class	G.grade
Bob	1	3.7	1	cs164	A
Al	4	4.0	4	cs61a	A
Al	4	4.0	4	cs170	A

Worksheet Part B

SELECT [DISTINCT] <column list>
FROM <relation list>
[**WHERE** <predicate>]
[**GROUP BY** <column list> [**HAVING** <predicate>]]
[**ORDER BY** <column list>];

The name of all songs with the genre "Country" which have spent more than 2 weeks in the top 40.

The name of all songs with the genre “Country” which have spent more than 2 weeks in the top 40.

```
SELECT Songs.song_name FROM Albums,
Songs WHERE Songs.album_id =
Albums.album_id AND Albums.genre =
'country' AND Songs.weeks_in_top_40
> 2;
```

For each song, its name, the name of its album, and the name of its artist.

For each song, its name, the name of its album, and the name of its artist.

```
SELECT Songs.song_name,
Albums.album_name,
Artists.artist_name FROM Artists,
Albums, Songs WHERE
Artists.artist_id =
Albums.artist_id AND Songs.album_id
= Albums.album_id;
```

The artist name and number of albums released by each artist.

The artist name and number of albums released by each artist.

```
SELECT Artists.artist_name,
COUNT(*) FROM Artists, Albums WHERE
Artists.artist_id =
Albums.artist_id GROUP BY
Artists.artist_id,
Artists.artist_name;
```

Find singers, with no duplicates, who released both “Techno” and “Pop” albums.

Find singers, with no duplicates, who released both “Techno” and “Pop” albums.

```
SELECT DISTINCT Artists.artist_name
FROM Artists, Albums A1, Albums A2
WHERE Artists.artist_id =
A1.artist_id AND A1.artist_id =
A2.artist_id AND A1.genre =
‘Techno’ AND A2.genre = ‘Pop’;
```

```
SELECT S.name
FROM Students S, Grades G
WHERE S.id = G.id AND G.grade = ‘A’;
```

```
SELECT S.name
FROM Students S INNER JOIN Grades G
ON S.id = G.id
WHERE G.grade = ‘A’;
```

```
CREATE TABLE dogs (
    dogid integer,
    owner integer,
    name text,
    breed text,
    age integer,
    PRIMARY KEY (dogid),
FOREIGN KEY (owner) REFERENCES users);
```

Which query(ies) give how many dog breeds are in your database?

A) SELECT COUNT(*) FROM dogs;

B) SELECT COUNT(*) FROM dogs GROUP BY name;

C) SELECT COUNT(name) FROM dogs;

D) SELECT COUNT(dogid) FROM dogs;

E) SELECT COUNT(DISTINCT breed) FROM dogs;

```
CREATE TABLE dogs (
    dogid integer,
    owner integer,
    name text,
    breed text,
    age integer,
PRIMARY KEY (dogid),
FOREIGN KEY (owner) REFERENCES users);
```

Which query(ies) give how many dog breeds are in your database?

A) SELECT COUNT(*) FROM dogs;

B) SELECT COUNT(*) FROM dogs GROUP BY name;

C) SELECT COUNT(name) FROM dogs;

D) SELECT COUNT(dogid) FROM dogs;

E) SELECT COUNT(DISTINCT breed) FROM dogs;

```
CREATE TABLE dogs (
 dogid integer,
 owner integer,
 name text,
 breed text,
 age integer,
 PRIMARY KEY (dogid),
 FOREIGN KEY (owner) REFERENCES users);
```

```
CREATE TABLE users (
 userid integer,
 name text,
 age integer,
PRIMARY KEY (userid));
```

Which query should you issue to find the user id of the user with the most dogs along with the number of dogs the user owns?

A)
```
SELECT userid, COUNT(*) as cnt
FROM dogs, users
WHERE userid = dogid
ORDER BY cnt DESC
LIMIT 1;
```

```
CREATE TABLE dogs (
 dogid integer,
 owner integer,
 name text,
 breed text,
 age integer,
 PRIMARY KEY (dogid),
 FOREIGN KEY (owner) REFERENCES users);
```

```
CREATE TABLE users (
 userid integer,
 name text,
 age integer,
PRIMARY KEY (userid));
```

Which query should you issue to find the user id of the user with the most dogs along with the number of dogs the user owns?

B)
```
SELECT userid, MAX(dogid) as max
FROM dogs,users
WHERE userid = owner
GROUP BY dogid, userid
ORDER BY max DESC
LIMIT 1;
```

```
CREATE TABLE dogs (
 dogid integer,
 owner integer,
 name text,
 breed text,
 age integer,
 PRIMARY KEY (dogid),
 FOREIGN KEY (owner) REFERENCES users);
```

```
CREATE TABLE users (
 userid integer,
 name text,
 age integer,
PRIMARY KEY (userid));
```

Which query should you issue to find the user id of the user with the most dogs along with the number of dogs the user owns?

C)

```
SELECT userid, MAX(dogid) as max
FROM dogs,users
WHERE userid = owner
GROUP BY userid
ORDER BY max DESC
LIMIT 1;
```

```
CREATE TABLE dogs (
 dogid integer,
 owner integer,
 name text,
 breed text,
 age integer,
 PRIMARY KEY (dogid),
 FOREIGN KEY (owner) REFERENCES users);
```

```
CREATE TABLE users (
 userid integer,
 name text,
 age integer,
 PRIMARY KEY (userid));
```

Which query should you issue to find the user id of the user with the most dogs along with the number of dogs the user owns?

D)

```
SELECT userid, COUNT(*) as cnt
FROM dogs,users
WHERE userid = owner
GROUP BY userid
ORDER BY cnt DESC
LIMIT 1;
```

```
CREATE TABLE dogs (
 dogid integer,
 owner integer,
 name text,
 breed text,
 age integer,
 PRIMARY KEY (dogid),
 FOREIGN KEY (owner) REFERENCES users);
```

```
CREATE TABLE users (
 userid integer,
 name text,
 age integer,
PRIMARY KEY (userid));
```

Which query should you issue to find the user id of the user with the most dogs along with the number of dogs the user owns?

D)

```
SELECT userid, COUNT(*) as cnt
FROM dogs,users
WHERE userid = owner
GROUP BY userid
ORDER BY cnt DESC
LIMIT 1;
```

```
Homes(home_id int, city text, bedrooms int,
      bathrooms int, area int)

Transactions(home_id int, buyer_id int, seller_id
      int, transaction_date date, sale_price int)

Buyers(buyer_id int, name text)

Sellers(seller_id int, name text)
```

Find the duplicate-free set of id's of all homes in Berkeley with at least 6 bedrooms and at least 2 bathrooms that were bought by "Bobby Tables".

```
Homes(home_id int, city text, bedrooms int,
      bathrooms int, area int)

Transactions(home_id int, buyer_id int, seller_id
      int, transaction_date date, sale_price int)

Buyers(buyer_id int, name text)

Sellers(seller_id int, name text)
```

Find the duplicate-free set of id's of all homes in Berkeley with at least 6 bedrooms and at least 2 bathrooms that were bought by "Bobby Tables".

```
SELECT DISTINCT H.home_id
FROM Homes H, Transactions T, Buyers B
WHERE H.home_id=T.home_id
AND T.buyer_id=B.buyer_id
AND H.city="Berkeley"
AND H.bedrooms>=6
AND H.bathrooms>=2
AND B.name="Bobby Tables";
```

```
Homes(home_id int, city text, bedrooms int,
      bathrooms int, area int)

Transactions(home_id int, buyer_id int, seller_id
      int, transaction_date date, sale_price int)

Buyers(buyer_id int, name text)

Sellers(seller_id int, name text)
```

Find the name of the city with the highest number of homes that have not yet been sold. Include the number of unsold homes in your output.

```
Homes(home_id int, city text, bedrooms int,
        bathrooms int, area int)

Transactions(home_id int, buyer_id int, seller_id
   int, transaction_date date, sale_price int)

Buyers(buyer_id int, name text)

Sellers(seller_id int, name text)
```

Find the name of the city with the highest number of homes that have not yet been sold. Include the number of unsold homes in your output.

```
SELECT H.city, COUNT(*) AS cnt
FROM Homes H
WHERE H.home_id NOT IN (
    SELECT DISTINCT T.home_id
    FROM Transactions T)
GROUP BY H.city
ORDER BY cnt DESC
LIMIT 1;
```