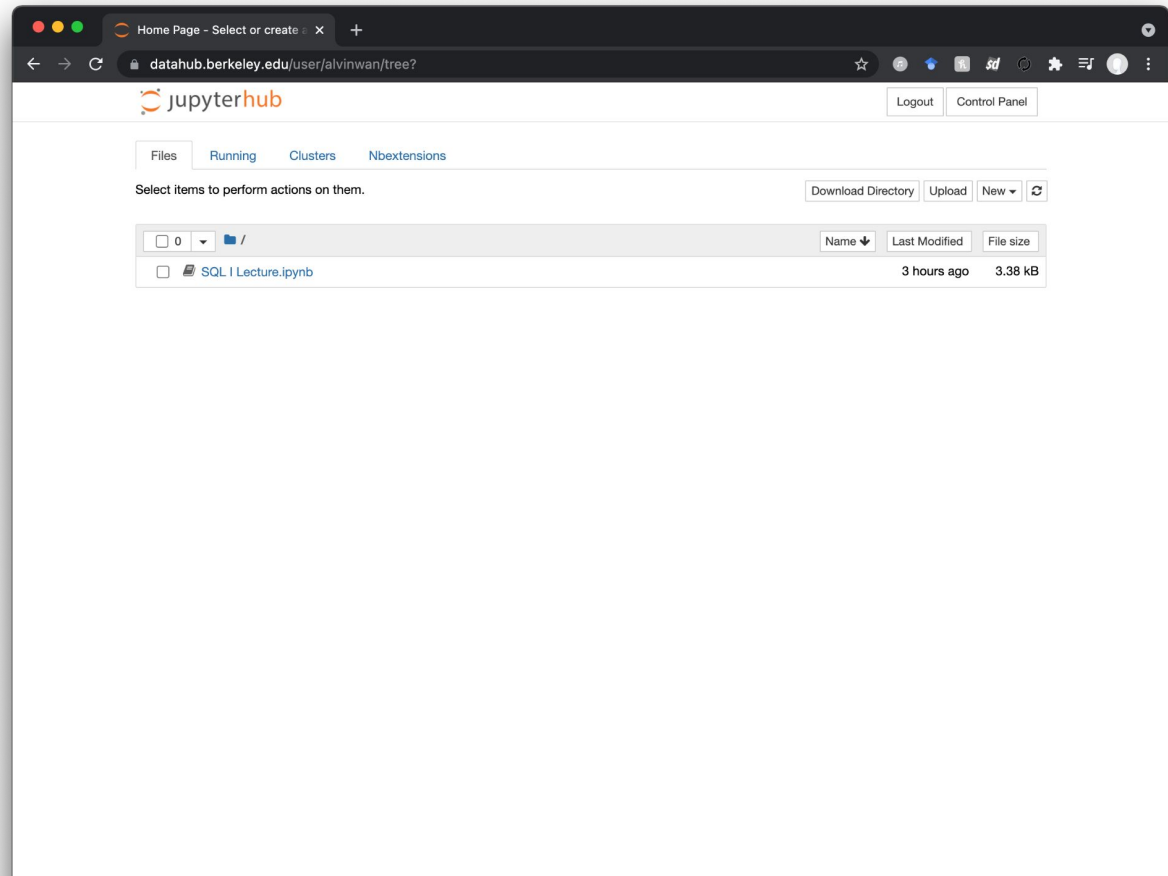
LECTURE 9

Join Tables with SQL Queries

Your first queries for multiple tables

Data 100/Data 200, Fall 2021 @ UC Berkeley
Fernando Pérez and Alvin Wan
(content by Alvin Wan, Anthony D. Joseph, Allen Shen, Josh Hug, John DeNero, Joseph Gonzalez)

datahub.berkeley.edu
notebook



CC BY NC SA

SQL I Lecture - Jupyter Notebo

datahub.berkeley.edu/user/alvinwan/notebooks/SQL%20I%

jupyterhub SQL I Lecture Last Checkpoint: La

File Edit View Insert Cell Kernel Widgets

Run Cod

```
In [2]: %load_ext sql

In [ ]:
```

CC BY SA

Breed

id	name
1	Corgi
2	Bernese
3	Bulldog

Pet

id	breed_id	name
1	1	Apricot
2	2	Boots

CC BY NC SA

Cross Join

Inner Join

Outer Join

Join Conditions

CC BY-NC-SA

Cross Join: All Pairs of Rows

```
SELECT *
FROM Pet AS p, Breed AS b;
```

Breed

id	name
1	Corgi
2	Bernese
3	Bulldog

Pet

id	breed_id	name
1	1	Apricot
2	2	Boots

b.id	b.name	p.id	p.breed_id	p.name
1	Corgi	1	1	Apricot
2	Bernese	1	1	Apricot
3	Bulldog	1	1	Apricot
1	Corgi	2	2	Boots
2	Bernese	2	2	Boots
3	Bulldog	2	2	Boots

Cross Join

Inner Join

Outer Join

Join Conditions

CC BY SA

Inner Join: Only Matching Rows

```
SELECT *
FROM Pet AS p
JOIN Breed AS b
    ON p.breed_id = b.id;
```

Breed

id	name
1	Corgi
2	Bernese
~~3~~	~~Bulldog~~

Pet

id	breed_id	name
1	1	Apricot
2	2	Boots

b.id	b.name	p.id	p.breed_id	p.name
1	Corgi	1	1	Apricot
2	Bernese	2	2	Boots

CC BY-NC-SA

Inner Join is Cross Join + Filter
(conceptually)

```
SELECT *
FROM Pet AS p, Breed AS b
WHERE p.breed_id = b.id;
```

Breed

id	name
1	Corgi
2	Bernese
3	Bulldog

Pet

id	breed_id	name
1	1	Apricot
2	2	Boots

b.id	b.name	p.id	p.breed_id	p.name
1	Corgi	1	1	Apricot
~~2~~	~~Bernese~~	~~1~~	~~1~~	~~Apricot~~
~~3~~	~~Bulldog~~	~~1~~	~~1~~	~~Apricot~~
~~1~~	~~Corgi~~	~~2~~	~~2~~	~~Boots~~
2	Bernese	2	2	Boots
~~3~~	~~Bulldog~~	~~2~~	~~2~~	~~Boots~~

CC BY SA

Cross Join

Inner Join

Outer Join

Join Conditions

CC BY SA

Left Outer Join: All Rows in 1st Table

```
SELECT *
FROM Breed as b
LEFT JOIN Pet AS p
    ON p.id = b.id;
```

Breed

id	name
1	Corgi
2	Bernese
3	Bulldog

Pet

id	breed_id	name
1	1	Apricot
2	2	Boots

b.id	b.name	p.id	p.breed_id	p.name
1	Corgi	1	1	Apricot
2	Bernese	2	2	Boots
3	Bulldog	NULL	NULL	NULL

CC BY NC SA

PRACTICAL TIP

Use `OUTER JOIN` to account for rows with no relationships.

Example: Report number of orders per user. `INNER JOIN` would omit users with 0 orders.

CC BY SA

Right Outer Join: All Rows in 2nd Table

NOT SUPPORTED IN SQLITE

```
SELECT *
FROM Pet AS p
RIGHT JOIN Breed AS b
    ON p.id = b.id;
```

Pet

id	name
~~1~~	~~Apricot~~
2	Boots
3	Cally
4	Eugene

Breed

id	name
2	persian
3	ragdoll
4	bengal
5	persian

p.id	p.name	b.id	b.name
2	Boots	2	persian
3	Cally	3	ragdoll
4	Eugene	4	bengal
		5	persian

Missing values are **NULL**.

CC BY SA

Full Outer Join: All Rows in Both Tables

NOT SUPPORTED IN SQLITE

```
SELECT *
FROM Pet AS p
FULL OUTER JOIN Breed AS b
    ON p.id = b.id;
```

Pet

id	name
1	Apricot
2	Boots
3	Cally
4	Eugene

Breed

id	name
2	persian
3	ragdoll
4	bengal
5	persian

p.id	p.name	b.id	b.name
1	Apricot		
2	Boots	2	persian
3	Cally	3	ragdoll
4	Eugene	4	bengal
		5	persian

Missing values are **NULL**.

CC BY SA

QUICK CHECK

Take a Student table and Signups table, which maps students to classes. Which join do you use, to report **the number of classes per student**?

A: Left Outer Join (or Right Outer Join. Full outer join is wasteful.) to account for students with 0 classes.

CC BY SA

QUICK CHECK

Take a Student table and Signups table, which maps students to classes. Which join do you use, to check **no student has exceeded the maximum 6 allowed classes**?

A: Inner Join, as we only look for students with >6 signups.

CC BY SA

Cross Join

Inner Join

Outer Join

Join Predicates

Join predicates don't have to be equality

```
SELECT *
FROM Student AS s
JOIN Teacher AS t
    ON s.age > t.age;
```

Student

age	name
29	Jameel
37	Jian
20	Emma

Teacher

age	name
52	Ira
27	John
36	Anuja

s.age	s.name	t.age	t.name
29	Jameel	27	John
37	Jian	27	John
37	Jian	36	Anuja

CC BY NC SA

QUICK CHECK

For each penguin, find all possible parents.

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

```
SELECT *
FROM Penguin AS child
JOIN Penguin AS parent
   ON child.age < parent.age;
```

```
SELECT *
FROM Penguin AS child, Penguin AS parent
WHERE child.age < parent.age;
```

CC BY SA

TAKEAWAY

Use **`OUTER JOIN`**s to account for rows with no relationships.

CC BY SA

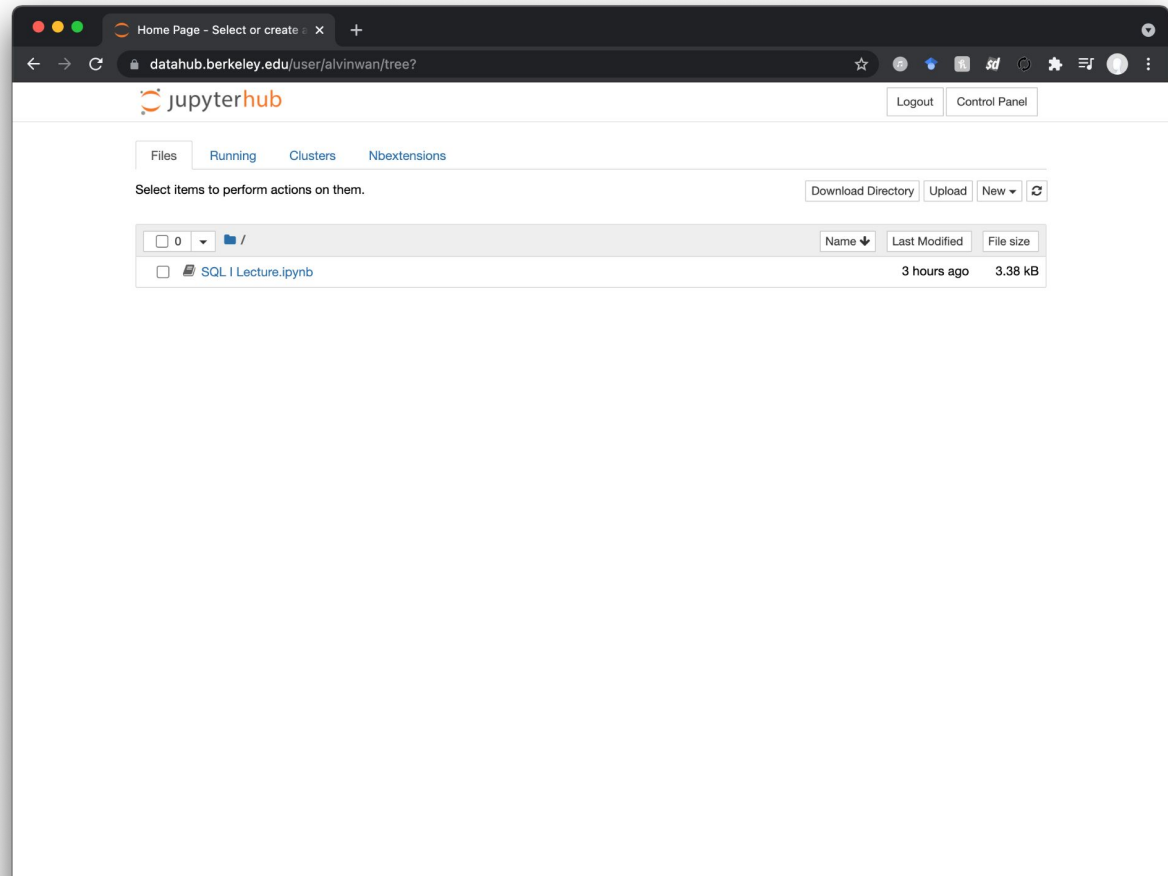
LECTURE 9

Handling **NULL** in **OUTER JOIN**s

Handling **NULL** in filters and aggregation

Data 100/Data 200, Fall 2021 @ UC Berkeley
Fernando Pérez and Alvin Wan
(content by Alvin Wan, Anthony D. Joseph, Allen Shen, Josh Hug, John DeNero, Joseph Gonzalez)

datahub.berkeley.edu
notebook



CC BY NC SA

SQL I Lecture - Jupyter Notebo

datahub.berkeley.edu/user/alvinwan/notebooks/SQL%20I%

jupyterhub SQL I Lecture Last Checkpoint: La

File Edit View Insert Cell Kernel Widgets

Run Cod

```
In [2]: %load_ext sql

In [ ]:
```

CC BY SA

Penguin

id	name	height
1	Alice	10
2	Bob	11
3	Cassie	8
4	NULL	NULL

CC BY NC SA

Comparators

Predicates

Aggregation

CC BY SA

Does “name = **NULL**” work?

❌
```
SELECT *
FROM Penguin
WHERE name = null;
```

Will always be empty

❌
```
SELECT name = null
FROM Penguin;
```

❌
```
SELECT name < null
FROM Penguin;
```

❌
```
SELECT name > null
FROM Penguin;
```

id	name	height
1	Alice	10
2	Bob	11
3	Cassie	8
4	NULL	NULL

CC BY-NC-SA

PRACTICAL TIP

Any comparator with `NULL` will result in `NULL`. Use `IS` predicate.

CC BY SA

Comparators

Predicates

Aggregation

CC BY SA

Use “name **IS NULL**”

✅
```
SELECT name IS NULL
FROM Penguin;
```

✅
```
SELECT name IS NOT NULL
FROM Penguin;
```

✅
```
SELECT *
FROM Penguin
WHERE name IS NOT NULL;
```

id	name	height
1	Alice	10
2	Bob	11
3	Cassie	8
4	NULL	NULL

CC BY SA

Comparators
Predicates
Aggregation

CC BY SA

Aggregates ignore **NULL**

✅
```
SELECT COUNT(name)
FROM Penguin;
```

✅
```
SELECT COUNT(*)
FROM Penguin;
```

✅
```
SELECT SUM(height)
FROM Penguin;
```

id	name	height
1	Alice	10
2	Bob	11
3	Cassie	8
4	NULL	NULL

CC BY-NC-SA

TAKEAWAY

When working with **`NULL`** in a predicate, use **`IS`**.

CC BY SA

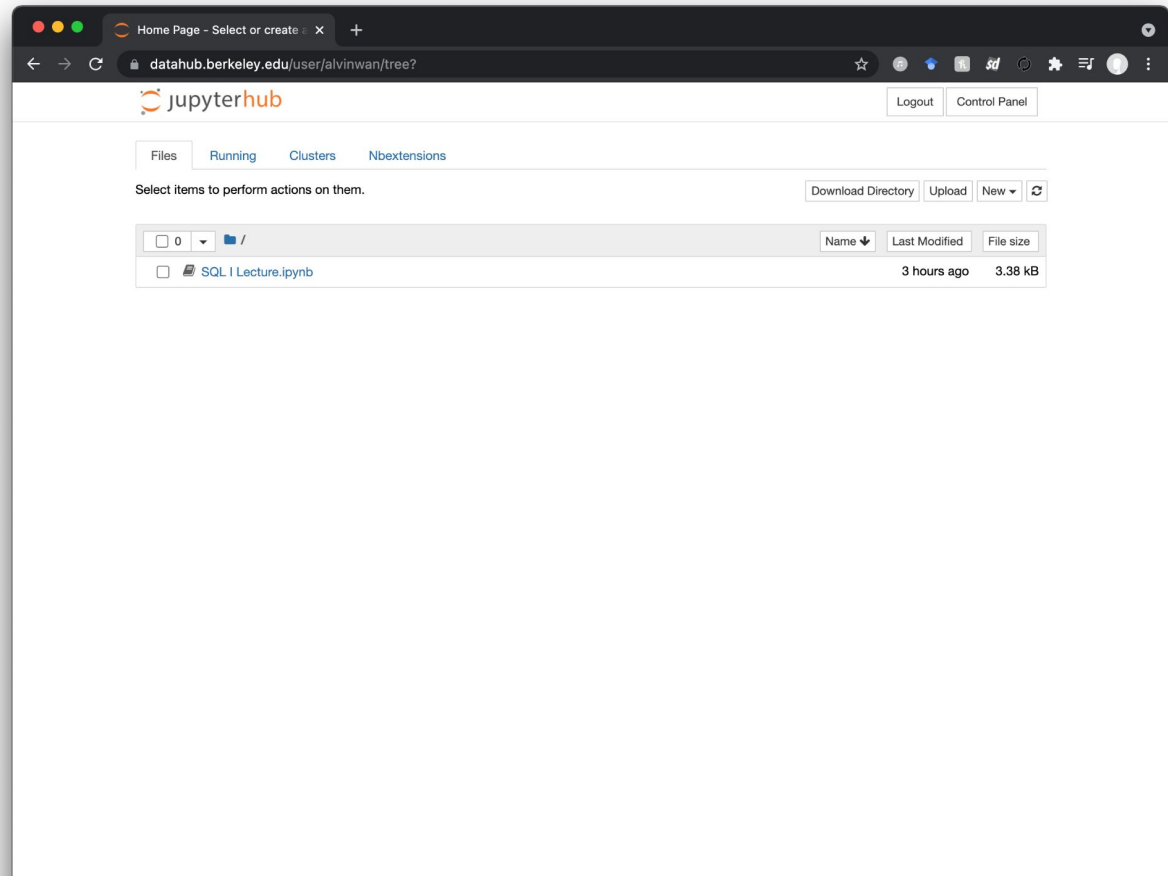
LECTURE 9

Subsample with SQL

Organizing your data

Data 100/Data 200, Fall 2021 @ UC Berkeley
Fernando Pérez and Alvin Wan
(content by Alvin Wan, Anthony D. Joseph, Allen Shen, Josh Hug, John DeNero, Joseph Gonzalez)

datahub.berkeley.edu
notebook



CC BY NC SA

SQL I Lecture - Jupyter Notebo

datahub.berkeley.edu/user/alvinwan/notebooks/SQL%20I%

jupyterhub SQL I Lecture Last Checkpoint: La

File Edit View Insert Cell Kernel Widgets

Run Cod

```
In [2]: %load_ext sql

In [ ]:
```

CC BY SA

Penguin

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

```
CREATE TABLE Penguin (
    id INT PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    height REAL CHECK WHERE (height >= 0),
    age INT NOT NULL
);
```

Subsample

Subsample By

Penguin

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

❌
```
SELECT * FROM Penguin LIMIT 3;
```

❌
```
SELECT * FROM Penguin
ORDER BY height
LIMIT 3;
```

✅
```
SELECT * FROM Penguin
ORDER BY RANDOM()
LIMIT 3;
```

CC BY SA

Subsample

Subsample By

Pick all penguins from 3 random ages.

Penguin

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

```
SELECT * FROM Penguin
WHERE age IN (
    SELECT age FROM Penguin
    GROUP BY age
    ORDER BY RANDOM()
    LIMIT 3
);
```

subquery

CC BY SA

PRACTICAL TIP

When you need multiple results from a subset of groups, use a subquery.

CC BY SA

Pick all penguins from 3 random ages.

Penguin

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

```
WITH ages AS (
    SELECT age FROM Penguin
    GROUP BY age
    ORDER BY RANDOM()
    LIMIT 3
)
SELECT * FROM Penguin
WHERE age IN ages;
```

Common Table Expression

CC BY SA

NOTE

Describe a one-column table for subqueries and CTEs used in a **`WHERE ... IN`** clause.

CC BY SA

QUICK CHECK

For 3 random penguins, show all possible parents.

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

id	name	height	age
1	Alice	3.6	10
2	Bob	4.0	15
3	Cassie	3.8	5
4	Dahlia	3.5	10
5	Eve	4.2	5
6	Fred	4.0	12
7	Glen	4.1	9

```
WITH children AS (
    SELECT id FROM Penguin
    ORDER BY RANDOM()
    LIMIT 3
)
SELECT *
FROM Penguin AS child
JOIN Penguin AS parent
    ON child.age < parent.age
WHERE child.id IN children;
```

CC BY SA

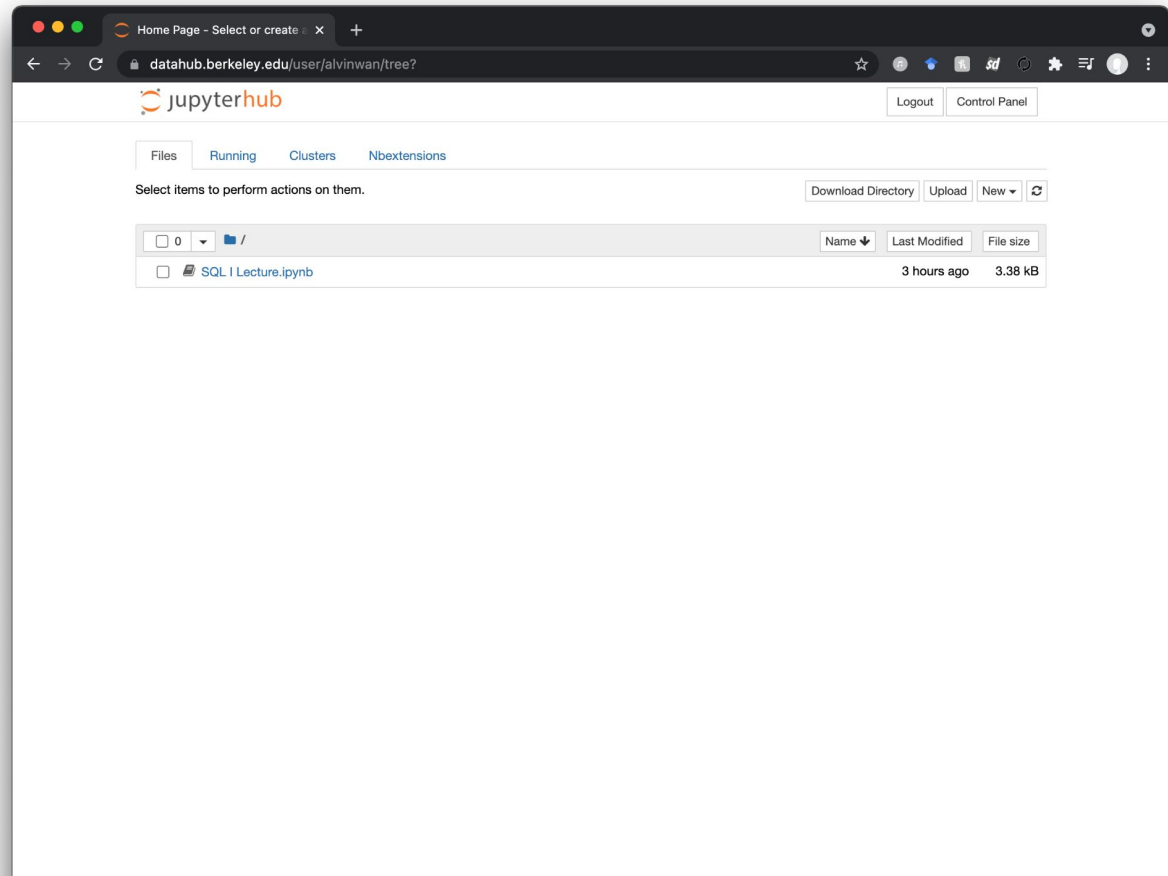
LECTURE 9

Practice

Write and Debug SQL Queries for Multiple Tables

Data 100/Data 200, Fall 2021 @ UC Berkeley
Fernando Pérez and Alvin Wan
(content by Alvin Wan, Anthony D. Joseph, Allen Shen, Josh Hug, John DeNero, Joseph Gonzalez)

datahub.berkeley.edu
notebook



CC BY NC SA

SQL I Lecture - Jupyter Notebo

datahub.berkeley.edu/user/alvinwan/notebooks/SQL%20I%

jupyterhub SQL I Lecture Last Checkpoint: La

File Edit View Insert Cell Kernel Widgets

Run Cod

```
In [2]: %load_ext sql

In [ ]:
```

CC BY SA

NOTE

CASE operates like a **SWITCH** statement, allowing you to rename values based on predicates.

Not just for prettification.
(We will see later)

```
CASE
    WHEN age >= 18 THEN 'adult'
    WHEN age > 12 AND age <= 18 THEN 'teen'
    WHEN age > 6 AND age <= 12 THEN 'kid'
    WHEN age <= 6 THEN 'toddler'
END AS 'category'
```

Only allowed in **SELECT**

CC BY SA

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

CC BY NC SA

Queries

Advanced Queries

Debug Queries

CC BY SA

For each dog, report name and lifespan.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT Dog.name, Breed.lifespan
FROM Dog
JOIN Breed
    ON Dog.breed_id = Breed.id;
```

CC BY SA

Report each dog's parent's name.

(exclude dogs without parent).

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT Child.name, Parent.name
FROM Dog as Child
JOIN Dog as Parent
    ON Parent.id = Child.parent_id;
```

CC BY SA

Report number of dogs per breed.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
--need to report all breeds, even
breed with 0 dogs!
SELECT Breed.name, COUNT(*)
FROM Breed
LEFT JOIN Dog
    ON Dog.breed_id = Breed.id
GROUP BY Breed.id;
```

name	COUNT(*)
Corgi	3
Bernese	1
Husky	1
Bulldog	1

Why is bulldog 1? There are no bulldogs.

CC BY-NC-SA

Report number of dogs per breed.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT Breed.name, COUNT(Dog.name)
FROM Breed
LEFT JOIN Dog
    ON Dog.breed_id = Breed.id
GROUP BY Breed.id;
```

CC BY-NC-SA

Queries

Advanced Queries

Debug Queries

CC BY SA

Report average height for each breed's children.

(Assume there are no grandparents)

1

2

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT
     Breed.name,
     AVG(Child.height)
FROM Breed
JOIN Dog AS Child
     ON Child.breed_id = Breed.id
JOIN Dog as Parent
     ON Child.parent_id = Parent.id
GROUP BY Breed.id;
```

1

2

CC BY SA

Report all possible playmates (same breed and within 1” height). Do not double-count pairs.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
--notice predicate is on foreign key. Don’t need to
condition on private keys!
--ensures each pair reported once
SELECT Dog1.name, Dog2.name
FROM Dog AS Dog1
JOIN Dog AS Dog2
     ON ABS(Dog1.height - Dog2.height) <= 1
     AND Dog1.breed_id = Dog2.breed_id
WHERE Dog1.id > Dog2.id;
```

CC BY-NC-SA

Report average height per dog family.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT
    AVG(Dog.height),
    CASE
        WHEN Dog.parent_id IS NULL THEN Dog.id
        ELSE Dog.parent_id
    END AS family_id
FROM Dog
GROUP BY family_id;
```

CC BY SA

Report parent name per dog family.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
--don’t get caught up the complexity!
Some queries can be solved simply.
SELECT name
FROM Dog
WHERE parent_id IS NULL;
```

CC BY SA

Report average height and parent name per dog family.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT
    AVG(Dog.height),
    CASE
        WHEN Dog.parent_id IS NULL THEN Dog.id
        ELSE Dog.parent_id
    END AS family_id,
     Parent.name
FROM Dog
JOIN Dog AS Parent
     ON family_id = Parent.id
GROUP BY family_id;
```

CC BY SA

Queries

Useful Queries

Debug Queries

CC BY SA

Report number of dogs per lifespan. What's wrong?

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT lifespan, COUNT(*),
FROM Breed
JOIN Dog
    ON Dog.breed_id = Breed.id
GROUP BY lifespan;
```

```
(sqlite3.OperationalError) near "FROM": syntax error
[SQL: SELECT lifespan, COUNT(*), FROM Breed
JOIN Dog
    ON Dog.breed_id = Breed.id
GROUP BY lifespan;]
```

CC BY-NC-SA

Report all dogs with names that start with 2 randomly-selected letters. What's wrong?

(Pick letters from names that actually exist.
Use `SUBSTR(COL, 1, 1)`)

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT name, SUBSTR(name, 1, 1) AS 'first'
FROM Dog
GROUP BY first
ORDER BY RANDOM()
LIMIT 2;
```

Maya missing!
Why?

name	first
Charlie	C
Maxie	M

Answer: Does not select all dogs from each sampled letter.

CC BY-NC-SA

Report all dogs with names that start with 2 randomly-selected letters. What's wrong?

(Pick letters from names that actually exist.
Use `SUBSTR(COL, 1, 1)`)

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
WITH Letter AS (
    SELECT SUBSTR(name, 1, 1) AS 'first'
    FROM Dog
    ORDER BY RANDOM()
    LIMIT 2
)
SELECT name
FROM Dog
WHERE SUBSTR(name, 1, 1) IN Letter;
```

Where's the 2nd letter?
Only has "M". Why?

name
Maxie
Maya

Answer: CTE samples 2 random rows, and selects first letter of name from random rows. Could potentially select both Ms.

CC BY-NC-SA

Report all dogs with names that start with 2 randomly-selected letters.

Answer

(Pick letters from names that actually exist.
Use `SUBSTR(COL, 1, 1)`)

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
WITH Letter AS (
    SELECT DISTINCT SUBSTR(name, 1, 1) AS 'first'
    FROM Dog
    ORDER BY RANDOM()
    LIMIT 2
)
SELECT name
FROM Dog
WHERE SUBSTR(name, 1, 1) IN Letter;
```

The CTE was sometimes picking “M” and “M”. Add DISTINCT constraint for letters.

CC BY-NC-SA

Report the tallest dog height for 2 random breeds. What's wrong?

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
WITH SampleBreed AS (
    SELECT id
    FROM Breed
    ORDER BY RANDOM()
    LIMIT 2
)
SELECT Dog.name, MAX(Dog.height)
FROM Dog
JOIN SampleBreed
     ON Dog.breed_id = SampleBreed.id
GROUP BY SampleBreed.id;
```

Try running this query a few times. Why is there sometimes only 1 breed returned?

name	MAX(Dog.height)
Apricot	11

CC BY-NC-SA

Report the tallest dog height for 2 random breeds. What's wrong?

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
WITH SampleBreed AS (
    SELECT Breed.id, Breed.name
    FROM Breed
    JOIN Dog
        ON Dog.breed_id = Breed.id
    GROUP BY Breed.id
    ORDER BY RANDOM()
    LIMIT 2
)
SELECT Dog.name, SampleBreed.name, MAX(Dog.height)
FROM Dog
JOIN SampleBreed
     ON Dog.breed_id = SampleBreed.id
GROUP BY SampleBreed.id;
```

Answer: The CTE is sometimes picking "Bulldog", which has no dogs.
Naive soln: Filter out any breed with no dogs.
Can we simplify?

CC BY-NC-SA

Report the tallest dog height for 2 random breeds. Answer

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT Breed.name, MAX(height)
FROM Dog
JOIN Breed
    ON Dog.breed_id = Breed.id
GROUP BY Breed.id
ORDER BY RANDOM()
LIMIT 2;
```

CC BY SA

TAKEAWAY

Get familiar with SQL **`JOIN`**s so you know what questions you can answer, and how to do it. Simpler is better.

CC BY SA

LECTURE 8

Practical Demo

Search NYC menus

Data 100/Data 200, Fall 2021 @ UC Berkeley
Fernando Pérez and Alvin Wan
(content by Alvin Wan, Anthony D. Joseph, Allen Shen, Josh Hug, John DeNero, Joseph Gonzalez)

Predicates

Casting

String Concatenation

Pandas

CC BY-NC-SA

BONUS: Of all child dogs, report the tallest one. How to make this query faster?

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT Child.name, Child.height
FROM Dog as Child
JOIN Dog as Parent
    ON Parent.id = Child.parent_id
ORDER BY Child.height desc
LIMIT 1;
```

CC BY-NC-SA

BONUS: Of all child dogs, report the tallest one. Faster query below.

Breed

id	name	lifespan
1	Corgi	15
2	Bernese	8
3	Husky	12
4	Bulldog	10

Dog

id	breed_id	name	height	parent_id
1	1	Apricot	11	NULL
2	2	Maxie	25	NULL
3	1	Charlie	8	1
4	3	Maya	24	1
5	1	Dixie	9	1

```
SELECT name, height
FROM Dog
WHERE parent_id IS NOT NULL
ORDER BY height DESC
LIMIT 1;
```

Replace JOIN with WHERE.

CC BY SA