Data100 Sp22 Disc 10
Probability/SQL

Attendance:

https://tinyurl.com/disc10michelle

Announcements

Due Dates

- Lab 10 due April 5th
- HW 7 due April 14th

Other

- Midterm on April 7

covers up to lecture 19

SQL

1

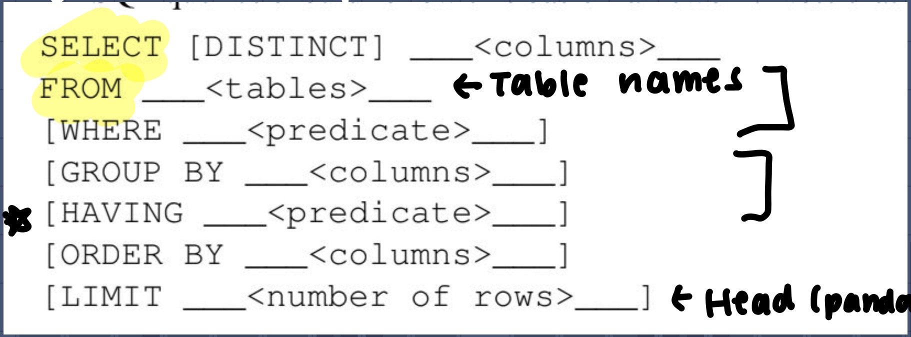
Syntax

Table

col 1 col 2 col 3

loc/ iloc | [ ]

column name



```
SELECT [DISTINCT] ___<columns>___
FROM ___<tables>___
[WHERE ___<predicate>___]
[GROUP BY ___<columns>___]
* [HAVING ___<predicate>___]
[ORDER BY ___<columns>___]
[LIMIT ___<number of rows>___]
```

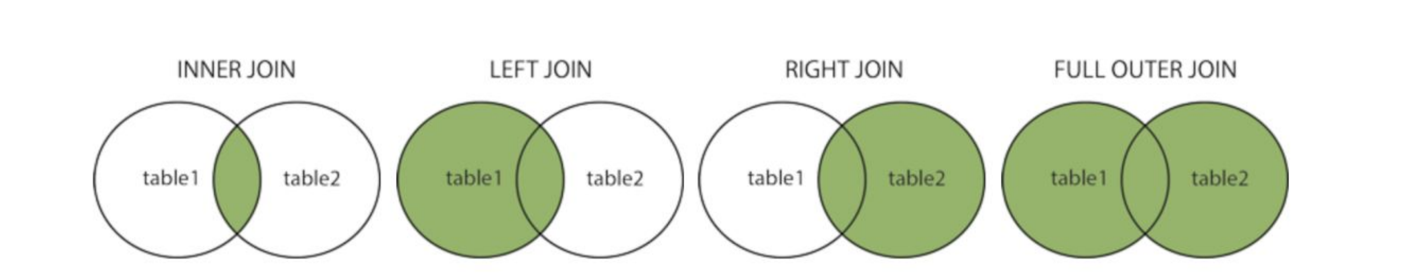
subset →

← Table names

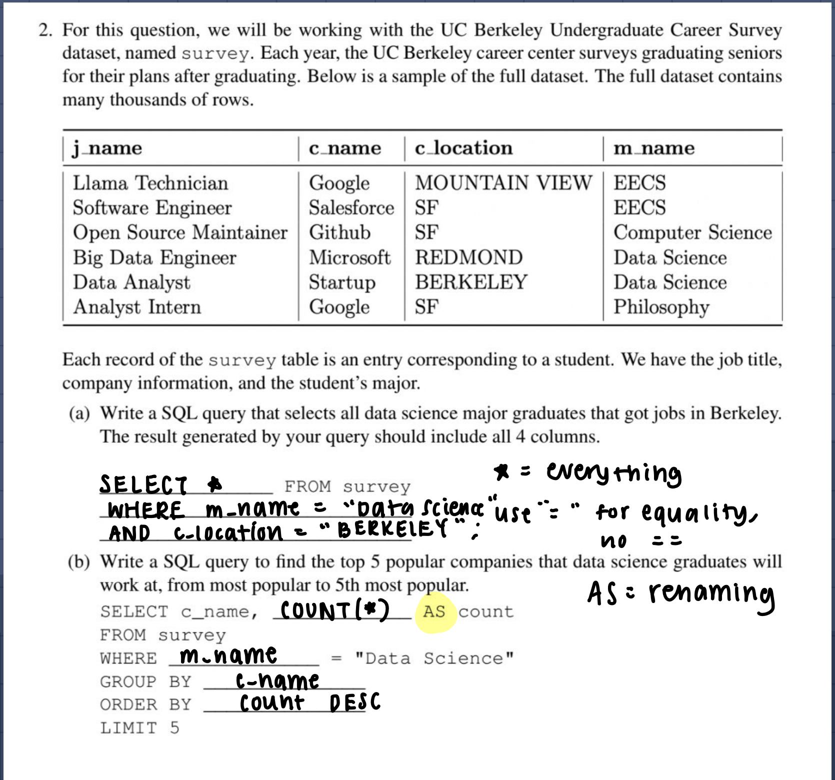
← Head (pandas)

similar to merge in Pandas

Joins



Q2



2. For this question, we will be working with the UC Berkeley Undergraduate Career Survey dataset, named `survey`. Each year, the UC Berkeley career center surveys graduating seniors for their plans after graduating. Below is a sample of the full dataset. The full dataset contains many thousands of rows.

j_name	c_name	c_location	m_name
Llama Technician	Google	MOUNTAIN VIEW	EECS
Software Engineer	Salesforce	SF	EECS
Open Source Maintainer	Github	SF	Computer Science
Big Data Engineer	Microsoft	REDMOND	Data Science
Data Analyst	Startup	BERKELEY	Data Science
Analyst Intern	Google	SF	Philosophy

Each record of the `survey` table is an entry corresponding to a student. We have the job title, company information, and the student's major.

(a) Write a SQL query that selects all data science major graduates that got jobs in Berkeley. The result generated by your query should include all 4 columns.

```
SELECT *  FROM survey
WHERE m_name = "Data Science"
AND c_location = "BERKELEY";
```

* = everything

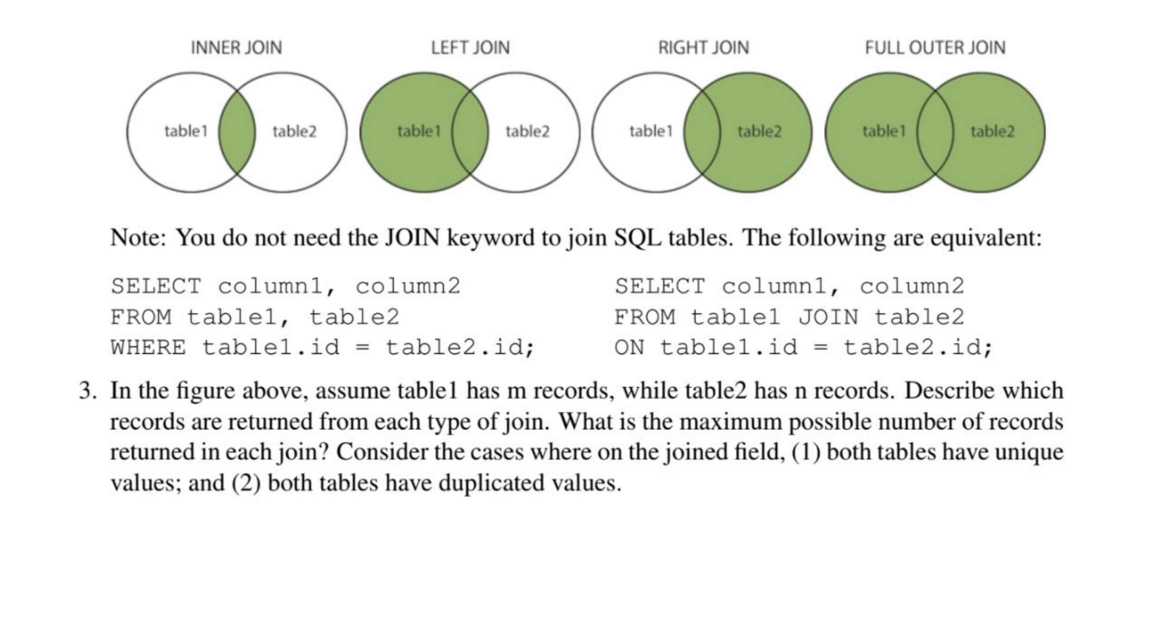
use "=" for equality, no ==

(b) Write a SQL query to find the top 5 popular companies that data science graduates will work at, from most popular to 5th most popular.

```
SELECT c_name, COUNT(*) AS count
FROM survey
WHERE m_name = "Data Science"
GROUP BY c_name
ORDER BY count DESC
LIMIT 5
```

AS: renaming

Q3



INNER JOIN
table1
table2
LEFT JOIN
table1
table2
RIGHT JOIN
table1
table2
FULL OUTER JOIN
table1
table2

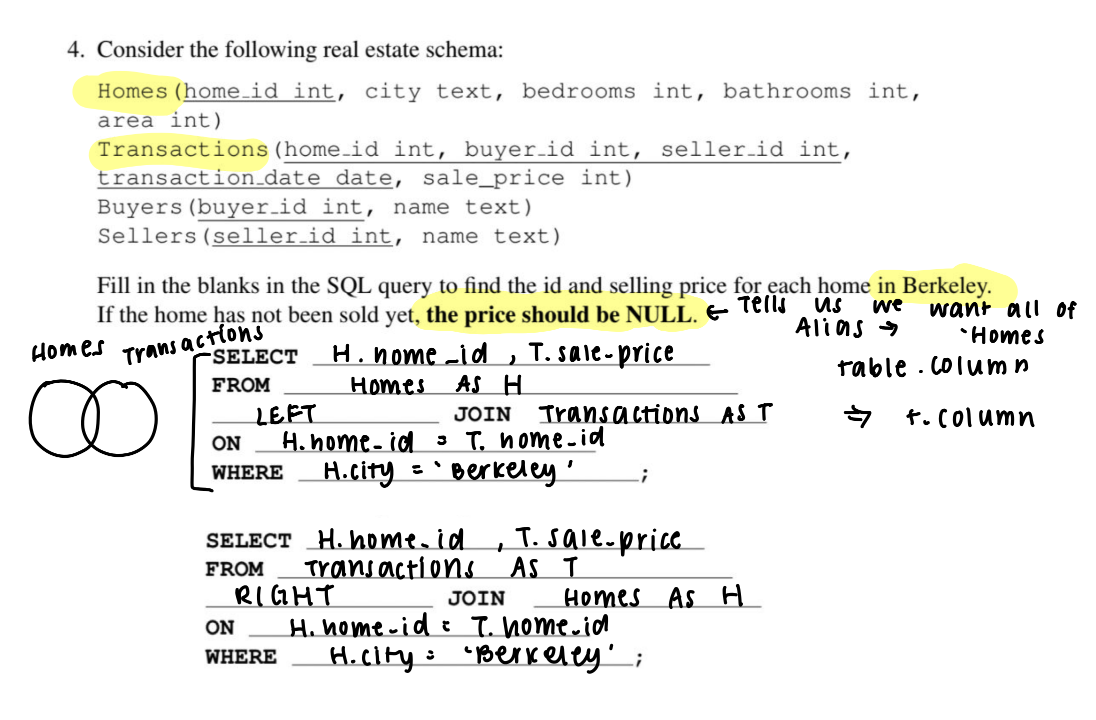
Note: You do not need the JOIN keyword to join SQL tables. The following are equivalent:

```
SELECT column1, column2
FROM table1, table2
WHERE table1.id = table2.id;
```

```
SELECT column1, column2
FROM table1 JOIN table2
ON table1.id = table2.id;
```

3. In the figure above, assume table1 has m records, while table2 has n records. Describe which records are returned from each type of join. What is the maximum possible number of records returned in each join? Consider the cases where on the joined field, (1) both tables have unique values; and (2) both tables have duplicated values.

Q4



4. Consider the following real estate schema:

```
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)
```

Fill in the blanks in the SQL query to find the id and selling price for each home in Berkeley. If the home has not been sold yet, **the price should be NULL.** ← Tells us we want all of `Homes

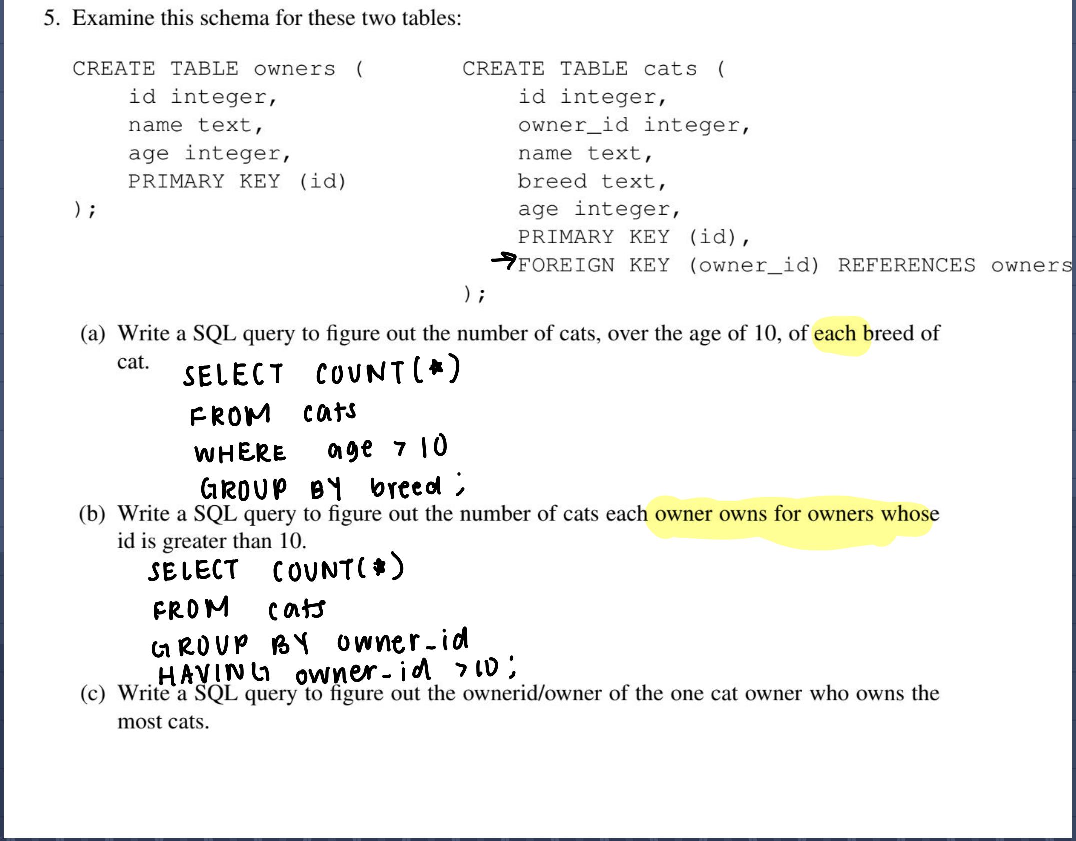
Alias → table.column ⇒ t.column

Homes Transactions

```
SELECT  H.home_id , T.sale-price
FROM    Homes AS H
   LEFT      JOIN  Transactions AS T
ON      H.home-id = T.home-id
WHERE   H.city = 'Berkeley' ;
```

```
SELECT  H.home-id , T.sale-price
FROM    Transactions AS T
   RIGHT      JOIN  Homes AS H
ON      H.home-id = T.home-id
WHERE   H.city = 'Berkeley' ;
```

Q5



5. Examine this schema for these two tables:

```
CREATE TABLE owners (
    id integer,
    name text,
    age integer,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE cats (
    id integer,
    owner_id integer,
    name text,
    breed text,
    age integer,
    PRIMARY KEY (id),
  → FOREIGN KEY (owner_id) REFERENCES owners
);
```

(a) Write a SQL query to figure out the number of cats, over the age of 10, of each breed of cat.

```
SELECT COUNT(*)
FROM cats
WHERE age > 10
GROUP BY breed;
```

(b) Write a SQL query to figure out the number of cats each owner owns for owners whose id is greater than 10.

```
SELECT COUNT(*)
FROM cats
GROUP BY owner_id
HAVING owner-id >10;
```

(c) Write a SQL query to figure out the ownerid/owner of the one cat owner who owns the most cats.