AI SATURDAYS LAGOS

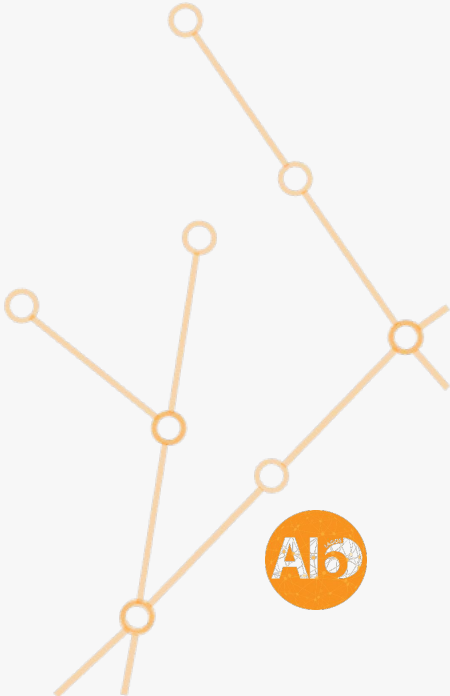
Week 4: Relational Data

Instructor: **'Tayo Jabar**

Outline

1. Relational Data
2. Entity Relationships
3. Manipulating Relational Data

   (Pandas & SQLite)
4. Joins



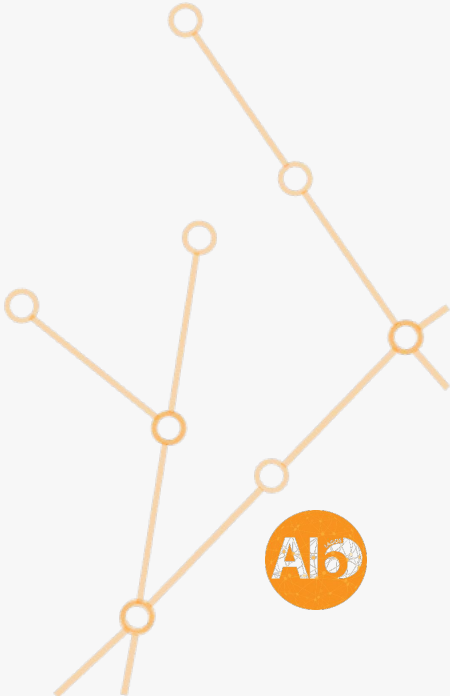
Relational Data

1. Overview
2. Primary & Foreign Keys
3. Indexes

Overview of Relational Data

Key word: Relation

- Organized, inter-related data
- Tabular structure
  - Rows
  - Columns
- Row: Record, Tuple
- Column: Feature, Attribute



Relational Data (...cont'd)

ID	First_Name	Last_Name	Gender	Email	Profession
101	Tayo	Jabar	Male	a@b.com	301
102	Kenechi	Dukor	Male	b@c.com	302
103	Tejumade	Afonja	Female	c@d.com	303
104	Fola	Animashaun	Female	d@e.com	302
105	Zazzu	Olalomi	Male	e@f.com	304

Legend

Row

Column

Primary Key

A **Primary Key** is a column/feature that is unique for every record in a table.

Primary & Foreign Keys

Customers

ID	First_Name	Last_Name	Gender	Email	Profession
1	Tayo	Jabar	Male	a@b.com	301
2	Kenechi	Dukor	Male	b@c.com	302
3	Tejumade	Afonja	Female	c@d.com	303
4	Fola	Animashaun	Female	d@e.com	302
5	Zazzu	Olalomi	Male	e@f.com	304

Professions

ID	Profession_Name
301	Doctor
302	Engineer
303	Professor
304	Mechanic

A **Foreign Key** is a primary key in one table that is used as an identifier in another table.

The ID column in the Professions table is a ***Primary Key*** in that table.

The Professions column in the Customers table is a ***Foreign Key***, identifying the professions in the Customers table.

Indexes

Location	ID	First_Name	Last_Name	Gender	Email	Profession
100	1	Tayo	Jabar	Male	a@b.com	301
200	2	Kenechi	Dukor	Male	b@c.com	302
300	3	Tejumade	Afonja	Female	c@d.com	303
400	4	Fola	Animashaun	Female	d@e.com	302
500	5	Zazzu	Olalomi	Male	e@f.com	304

Last Name	Location
Afonja	300
Animashaun	400
Dukor	200
Jabar	100
Olalomi	500

An Index is a database structure used to help speed up searching for an item in a database table.

An Index Table is a separate “sorted table” that holds the location of records within a table.

Quiz

Question A: *What is a relational database?*

a) A database that plays videos

b) A database that uses tables to store and organize data

c) A database that only works on mobile devices

d) A database for music streaming

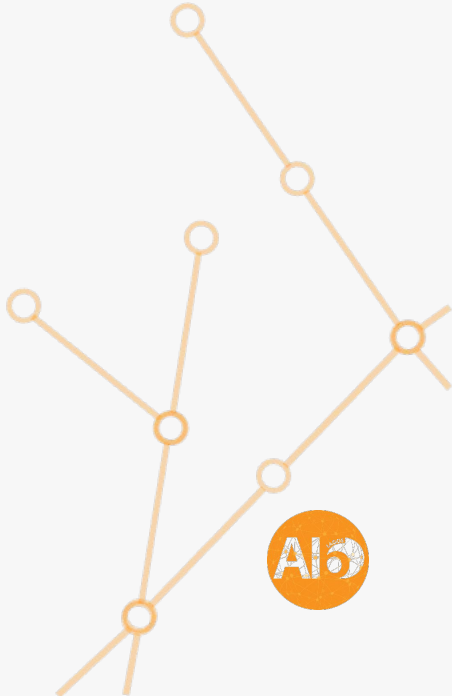
Question B: *What does the primary key in a relational database table represent?*

a) The main table in the database

b) A unique identifier for each row in a table

c) A way to delete data from a table

d) A foreign key in another table

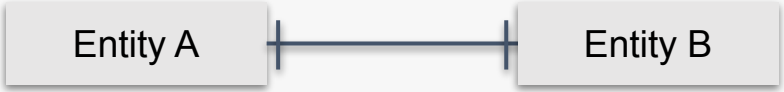


Entity Relationships

1. One to One
2. One to Many
3. Many to Many

One to One

- This shows an exclusive/strict case of one record in one table only matching one record in a second table.



Customers

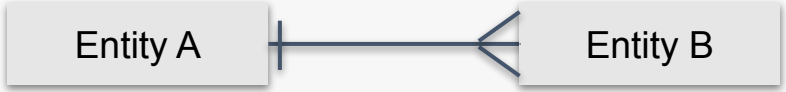
ID	First_Name	Last_Name	Gender	Email	Profession
1	Tayo	Jabar	Male	a@b.com	301
2	Kenechi	Dukor	Male	b@c.com	302
3	Tejumade	Afonja	Female	c@d.com	303
4	Fola	Animashaun	Female	d@e.com	302
5	Zazzu	Olalomi	Male	e@f.com	304

Usernames

Customer ID	Username
1	tayo_jabar
2	kenny_dukor
3	teju_afonja
4	fola_animashaun
5	zazzu_olalomi

One to Many / Many to One

- This illustrates a case where one record in one table could match multiple records in another table or vice-versa.



Professions

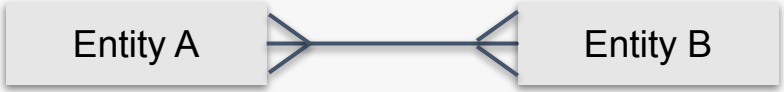
ID	Profession_Name
301	Doctor
302	Engineer
303	Professor
304	Mechanic

Customers

ID	First_Name	Last_Name	Gender	Email	Profession
1	Tayo	Jabar	Male	a@b.com	301
2	Kenechi	Dukor	Male	b@c.com	302
3	Tejumade	Afonja	Female	c@d.com	303
4	Fola	Animashaun	Female	d@e.com	302
5	Zazzu	Olalomi	Male	e@f.com	304

Many to Many

- This illustrates a case where multiple records in one table could match multiple records in another table or vice-versa.



Customers

ID	First_Name	Last_Name	Gender	Email	Profession
1	Tayo	Jabar	Male	a@b.com	301
2	Kenechi	Dukor	Male	b@c.com	302
3	Tejumade	Afonja	Female	c@d.com	303
4	Fola	Animashaun	Female	d@e.com	302
5	Zazzu	Olalomi	Male	e@f.com	304

Items

ID	Items
531	Tissues
532	Samosa
533	Pens
534	Airpods

Many to Many: Associative Tables

ID	First_Name	Last_Name	Gender	Email	Profession
1	Tayo	Jabar	Male	a@b.com	301
2	Kenechi	Dukor	Male	b@c.com	302
3	Tejumade	Afonja	Female	c@d.com	303
4	Fola	Animashaun	Female	d@e.com	302
5	Zazzu	Olalomi	Male	e@f.com	304

ID	Items
531	Tissues
532	Samosa
533	Pens
534	Airpods

Associative Table:
Items Bought / Sales

ID	Customer	Items
1101	101	531
1102	102	531
1103	103	533
1104	103	534
1105	104	532
1106	105	533
1107	102	534

Quiz

Question A: *What does an entity represent in entity-relationship modeling?*

a) A computer program

b) A real-world object or concept

c) A mathematical equation

d) A data type in Python

Question B: *Which of the following best describes a foreign key in entity-relationship modeling?*

a) A key that unlocks a treasure chest

b) A key used to access the internet

c) A key used to establish relationships between entities

d) A key that opens a physical door

Quiz

Question C: In a many-to-many relationship, what is typically used to represent the relationship in a relational database?

a) A foreign key in one of the related tables

b) A new table that connects the two entities

c) A primary key in each of the related tables

d) A special "many-to-many" keyword

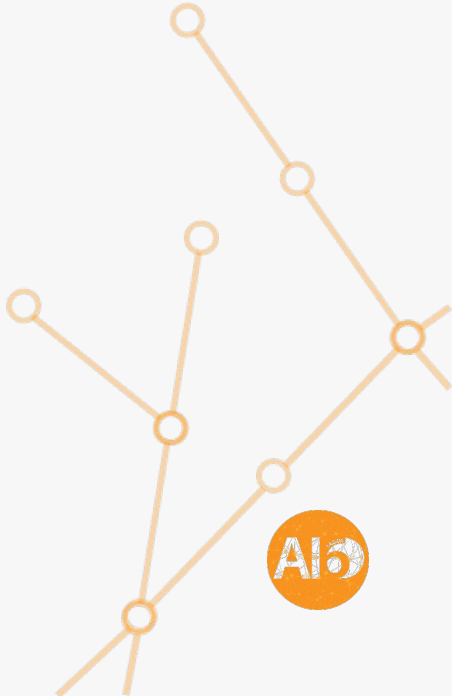
Question D: What is an associative entity in an entity-relationship diagram?

a) An entity that represents a physical object

b) An entity with a weak relationship to other entities

c) An entity that connects two or more other entities

d) An entity with no attributes



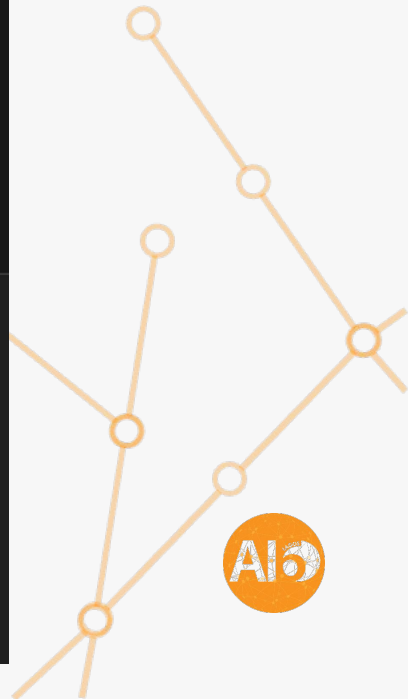
Manipulating Relational Data

1. Pandas
2. SQLite

Pandas

```
import pandas as pd

df = pd.DataFrame([(1, 'Kolter', 'Zico'),
                   (2, 'Xi', 'Edgar'),
                   (3, 'Lee', 'Mark'),
                   (4, 'Mani', 'Shouvik'),
                   (5, 'Gates', 'Bill'),
                   (6, 'Musk', 'Elon')],
                  columns=["id", "last_name", "first_name"])
df
```

	id	last_name	first_name
0	1	Kolter	Zico
1	2	Xi	Edgar
2	3	Lee	Mark
3	4	Mani	Shouvik
4	5	Gates	Bill
5	6	Musk	Elon



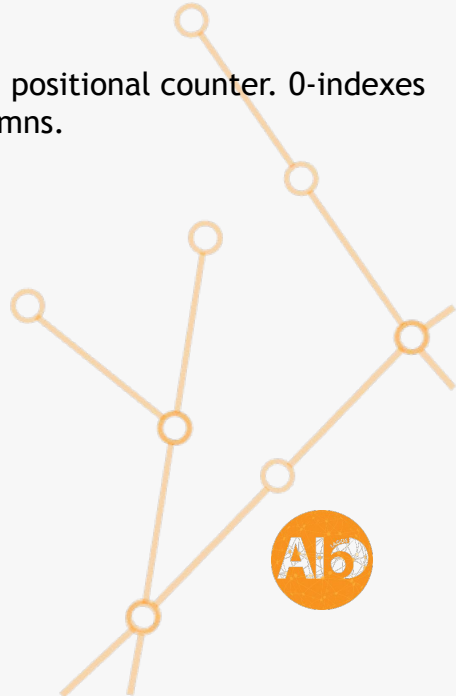
```
print (df.loc[1, "last_name"])
print ("######")
print (df.loc[:, "last_name"])
print ("######")
print (df.loc[:, ["last_name"]])
print ("######")

print (df.iloc[4,1])
print (df.loc[4, "last_name"])
```

✓ 0.0s

```
Kolter
######
id
1    Kolter
2        Xi
3       Lee
4      Mani
5     Gates
6      Musk
Name: last_name, dtype: object
######
   last_name
id
1     Kolter
2         Xi
3        Lee
4       Mani
5      Gates
6       Musk
######
Bill
Mani
```

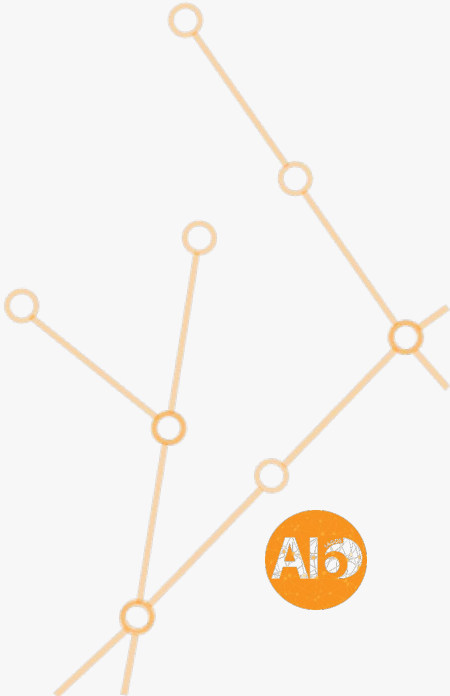
`df.loc:` based upon the "index" (i.e., effectively primary key) of the data frame.

`df.iloc:` based on the 0-indexed positional counter. 0-indexes can be used for both rows and columns.



SQLite

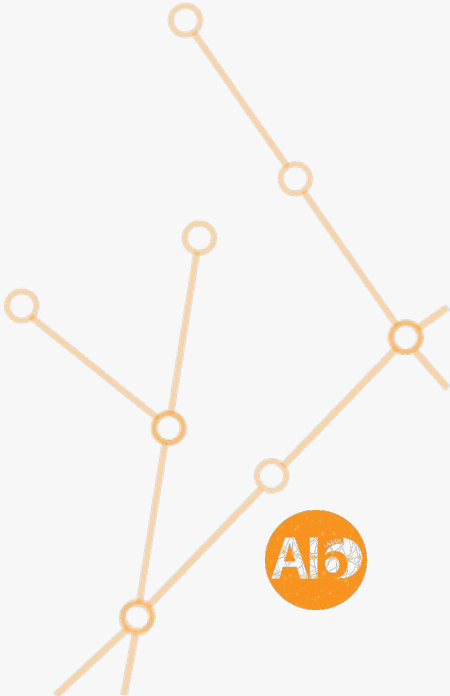
- A relational database management system (RDBMS)
- Direct connection to a database file. It is li(*gh*)te and is present on the edge
- Structured Query Language (SQL)
- Transactions maintain the ACID properties
  - Atomicity
  - Consistency
  - Isolation
  - Durability



SQLite (…cont'd)

```
import sqlite3
conn = sqlite3.connect("database.db")
cursor = conn.cursor()

### when you are done, call conn.close()
```

```
cursor.execute("""
CREATE TABLE person (
    id INTEGER PRIMARY KEY,
    last_name TEXT,
    first_name TEXT
);""")

cursor.execute("""
CREATE TABLE grades (
    person_id INTEGER PRIMARY KEY,
    hw1_grade INTEGER,
    hw2_grade INTEGER
);""")

conn.commit()
```



```
cursor.execute("INSERT INTO person VALUES (1, 'Kolter', 'Zico');")
cursor.execute("INSERT INTO person VALUES (2, 'Xi', 'Edgar');")
cursor.execute("INSERT INTO person VALUES (3, 'Lee', 'Mark');")
cursor.execute("INSERT INTO person VALUES (4, 'Mani', 'Shouvik');")
cursor.execute("INSERT INTO person VALUES (5, 'Gates', 'Bill');")
cursor.execute("INSERT INTO person VALUES (6, 'Musk', 'Elon');")

cursor.execute("INSERT INTO grades VALUES (5, 85, 95);")
cursor.execute("INSERT INTO grades VALUES (6, 80, 60);")
cursor.execute("INSERT INTO grades VALUES (100, 100, 100);")
```

Reading from an SQLite table

SELECT <columns> FROM <tables> WHERE <conditions>

```
for row in cursor.execute("SELECT * FROM person;"):
    print(row)
```

```
(1, 'Kolter', 'Zico')
(2, 'Xi', 'Edgar')
(3, 'Lee', 'Mark')
(4, 'Mani', 'Shouvik')
(5, 'Gates', 'Bill')
(6, 'Musk', 'Elon')
```

```
pd.read_sql_query("SELECT * from person;", conn, index_col="id")
```

id	last_name	first_name
1	Kolter	Zico
2	Xi	Edgar
3	Lee	Mark
4	Mani	Shouvik
5	Gates	Bill
6	Musk	Elon

We Value Your Feedback.

https://docs.google.com/forms/d/e/1FAIpQLSfoYXRTHJfjQo1Xj2PrgoNLKWOAUeiVl4MLrRrOACtYFzM8cg/viewform?usp=pp_url

Joins

1. Inner Join
2. Left Join
3. Right Join
4. Outer Join

SQL Joins

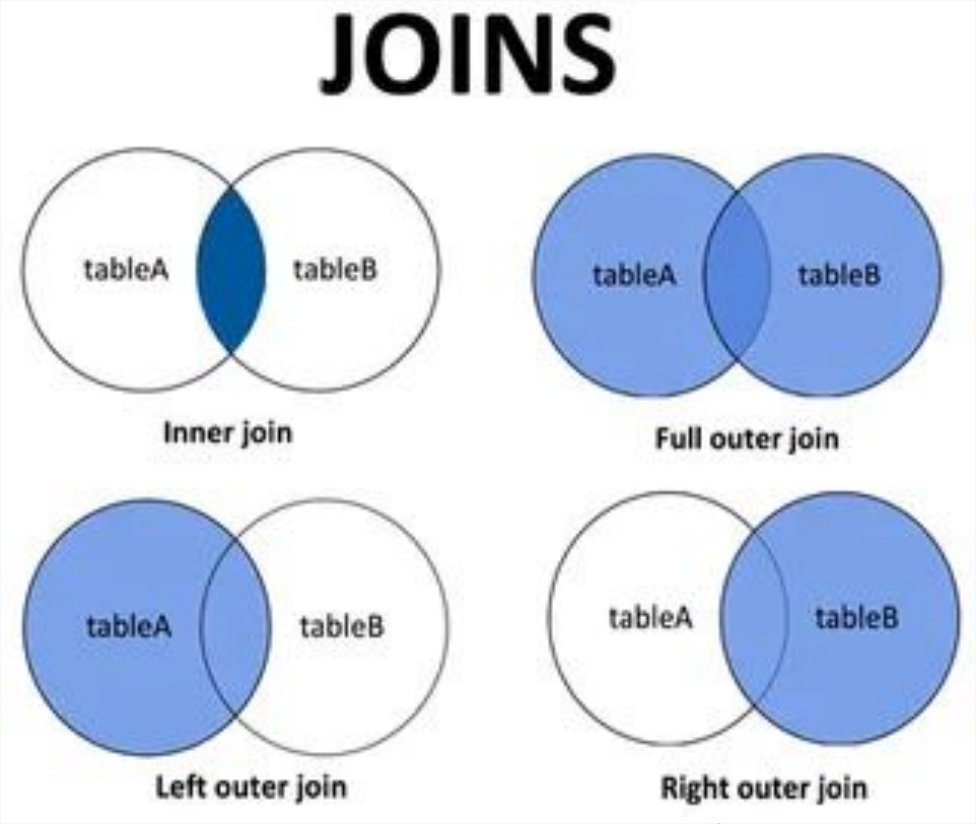


Image from Numpy Ninja

Given the following tables:

Person

id	last_name	first_name
1	Kolter	Zico
2	Xi	Edgar
3	Lee	Mark
4	Mani	Shouvik
5	Gates	Bill
6	Musk	Elon

Grades

person_id	hw1_grade	hw2_grade
5	85	95
6	80	60
100	100	100

Inner Join

Using Pandas

```
df_person = pd.read_sql_query("SELECT * FROM person", conn)
df_grades = pd.read_sql_query("SELECT * FROM grades", conn)
df_person.merge(df_grades, how="inner", left_on = "id", right_on="person_id")
```

	id	last_name	first_name	person_id	hw1_grade	hw2_grade
0	5	Gates	Bill	5	85	95
1	6	Musk	Elon	6	80	60

Using SQL

```
pd.read_sql_query("SELECT * FROM person, grades WHERE person.id = grades.person_id" , conn)
```

	id	last_name	first_name	person_id	hw1_grade	hw2_grade
0	5	Gates	Bill	5	85	95
1	6	Musk	Elon	6	80	60

Left Join

Using Pandas

```
df_person = pd.read_sql_query("SELECT * FROM person", conn)
df_grades = pd.read_sql_query("SELECT * FROM grades", conn)
df_person.merge(df_grades, how="left", left_on = "id", right_on="person_id")
```

	id	last_name	first_name	person_id	hw1_grade	hw2_grade
0	1	Kolter	Zico	NaN	NaN	NaN
1	2	Xi	Edgar	NaN	NaN	NaN
2	3	Lee	Mark	NaN	NaN	NaN
3	4	Mani	Shouvik	NaN	NaN	NaN
4	5	Gates	Bill	5.0	85.0	95.0
5	6	Musk	Elon	6.0	80.0	60.0

Using SQL

```
pd.read_sql_query("SELECT * FROM person LEFT JOIN grades ON person.id = grades.person_id" , conn)
```

Right Join

Using Pandas

```
df_person.merge(df_grades, how="right", left_on = "id", right_on="person_id")
```

	id	last_name	first_name	person_id	hw1_grade	hw2_grade
0	5.0	Gates	Bill	5	85	95
1	6.0	Musk	Elon	6	80	60
2	NaN	NaN	NaN	100	100	100

Using SQL

```
pd.read_sql_query("SELECT * FROM grades LEFT JOIN person ON grades.person_id = person.id" , conn)
```

✓ 0.0s

	person_id	hw1_grade	hw2_grade	id	last_name	first_name
0	5	85	95	5.0	Gates	Bill
1	6	80	60	6.0	Musk	Elon
2	100	100	100	NaN	None	None

Outer Join

```
df_person.merge(df_grades, how="outer", left_on = "id", right_on="person_id")
```

	id	last_name	first_name	person_id	hw1_grade	hw2_grade
0	1.0	Kolter	Zico	NaN	NaN	NaN
1	2.0	Xi	Edgar	NaN	NaN	NaN
2	3.0	Lee	Mark	NaN	NaN	NaN
3	4.0	Mani	Shouvik	NaN	NaN	NaN
4	5.0	Gates	Bill	5.0	85.0	95.0
5	6.0	Musk	Elon	6.0	80.0	60.0
6	NaN	NaN	NaN	100.0	100.0	100.0

Quiz

Question A: *In an SQL LEFT JOIN, which table's data is preserved entirely?*

a) Left table

b) Right table

c) Both tables

d) None of the tables

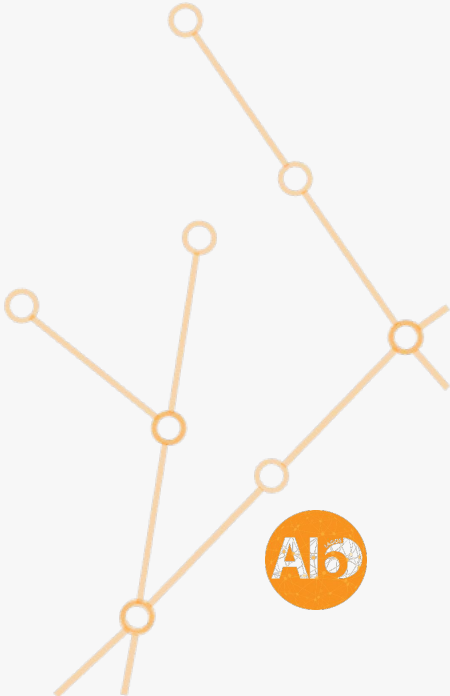
Question B: *What is the primary criterion for joining tables in a SQL query?*

a) The number of rows in each table

b) The data types of the columns

c) A common column or key between the tables

d) The order of the columns



Thank You

Resources and External Links

- Practical Data Science: Relational Data - CMU AI, Zico Kolter, Pat Virtue
  - https://www.datasciencecourse.org/slides/15388_S22_Lecture_4_relational_data.pdf
  - http://www.datasciencecourse.org/notes/relational_data
- What is a Relational Database? - IBM, Relational Databases
- SQLite, https://www.sqlite.org/index.html