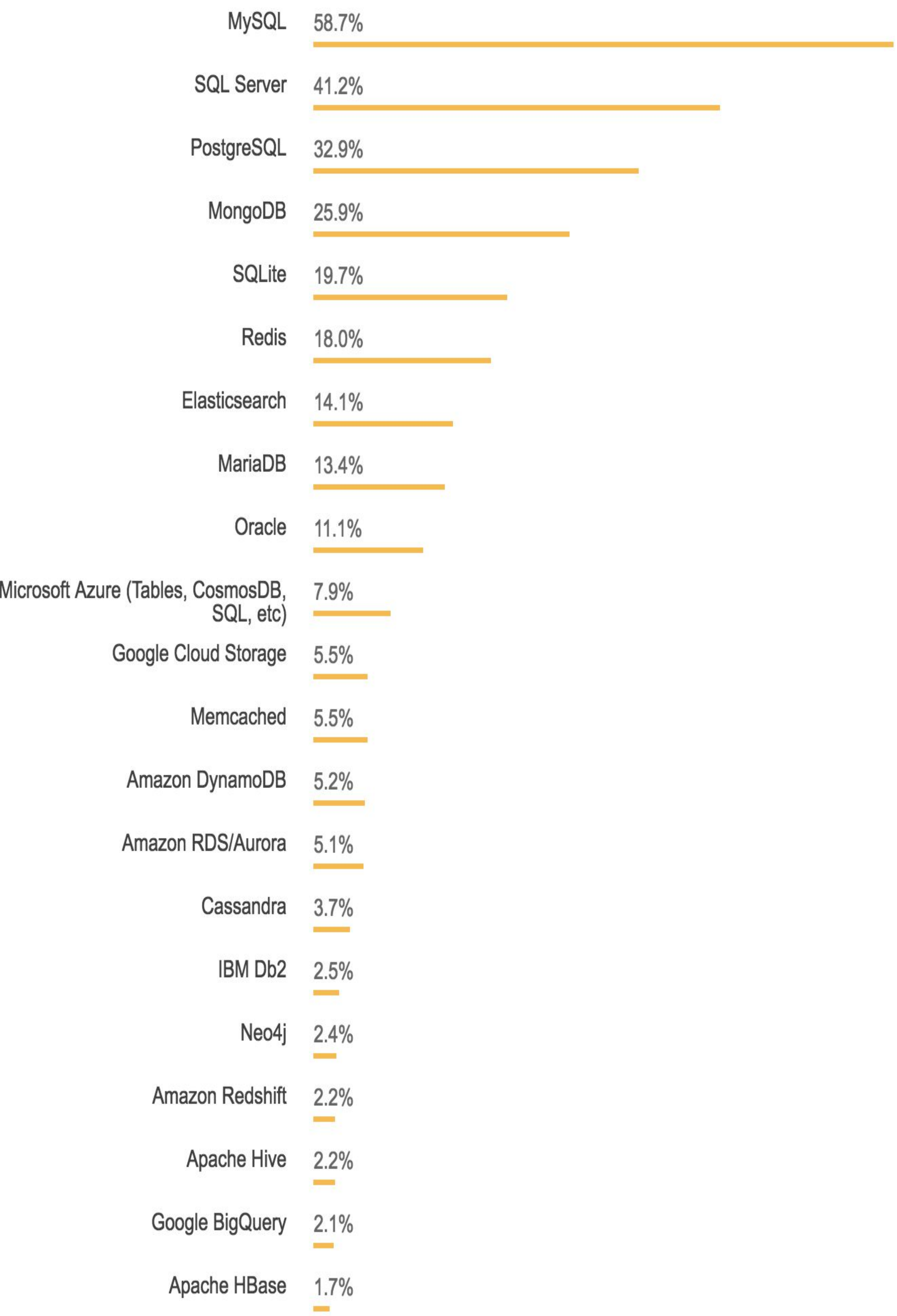
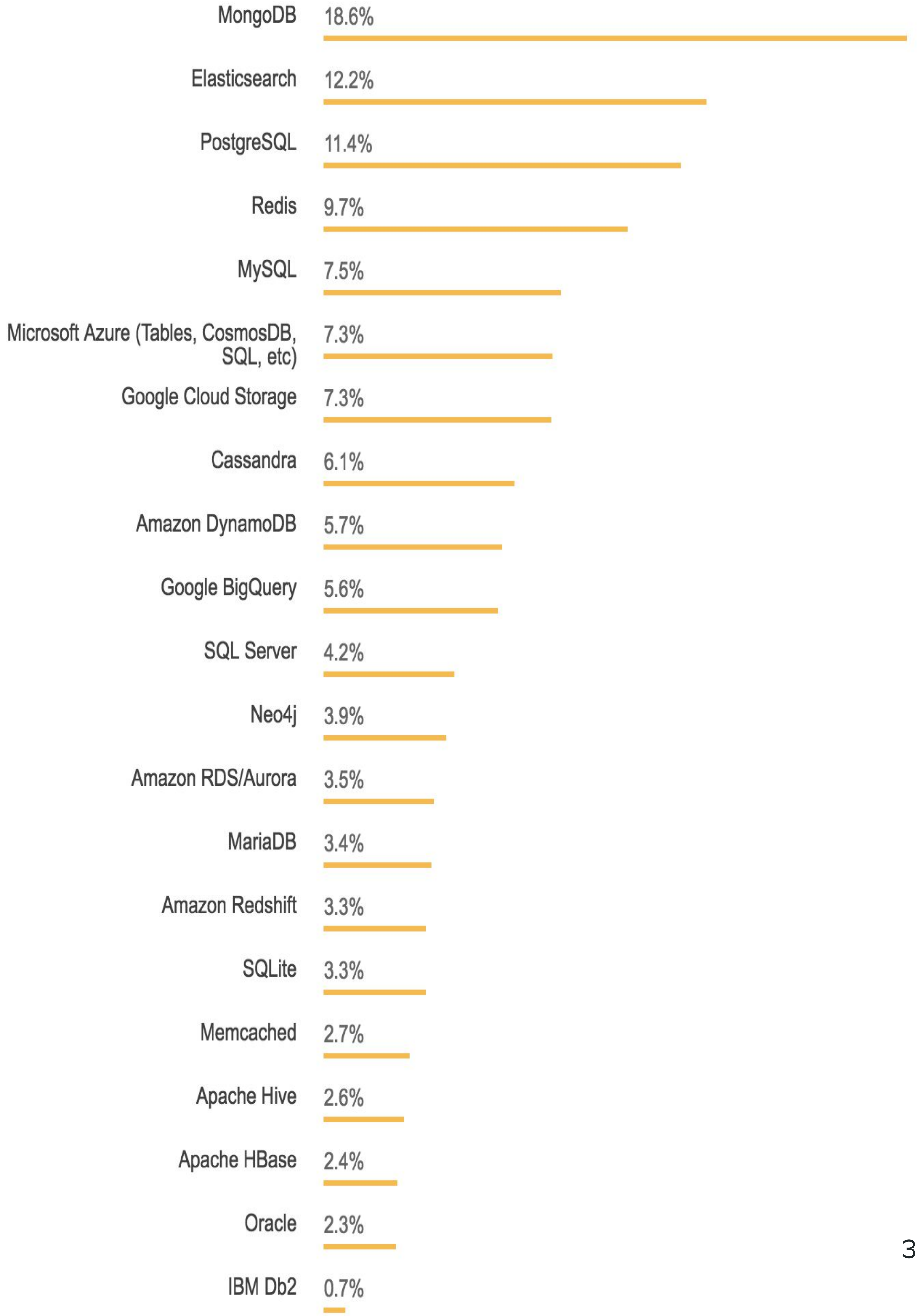
Introduction to Database

MySQL



Most used database . (SO)

Most wanted database . (SO)



Relational Database : Eg - SQL

SQL databases use structured query language (SQL) for defining and manipulating data. On one hand, this is extremely powerful:

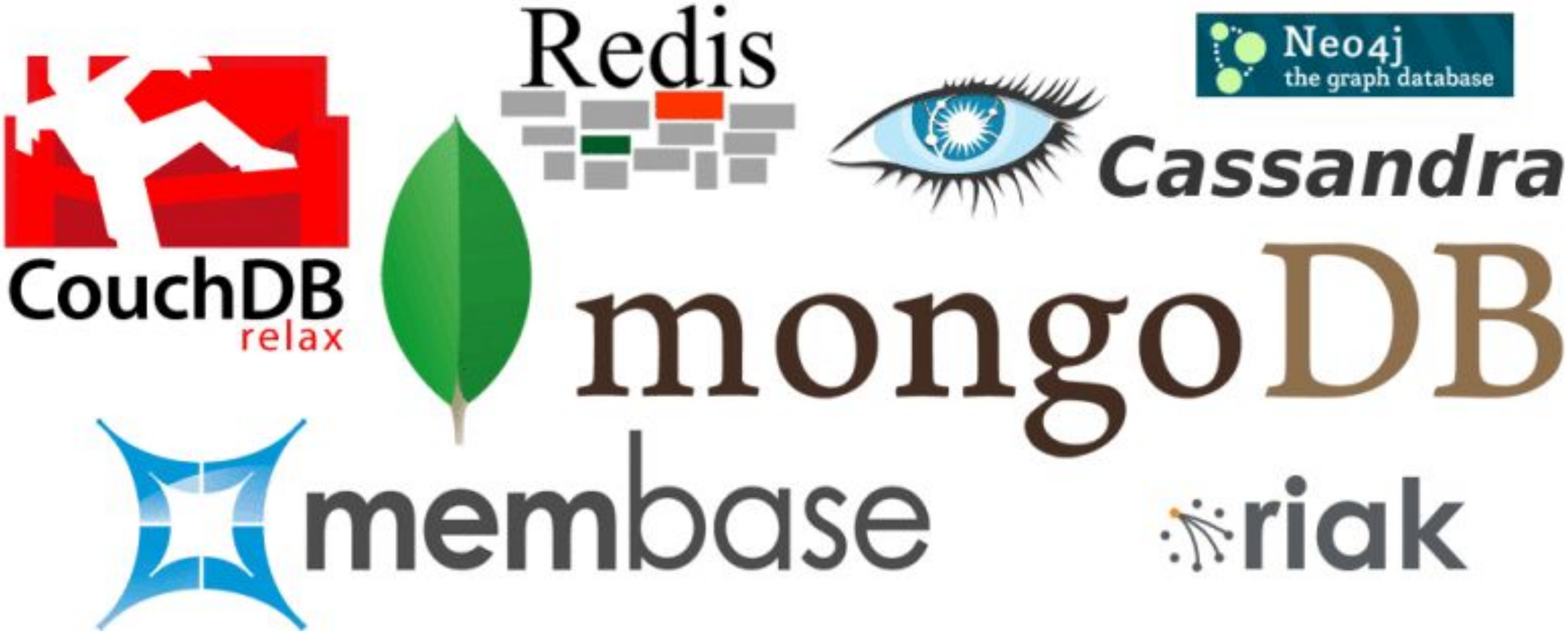
The most versatile and widely-used options available, making it a safe choice and especially great for complex queries.

Structure is restrictive.

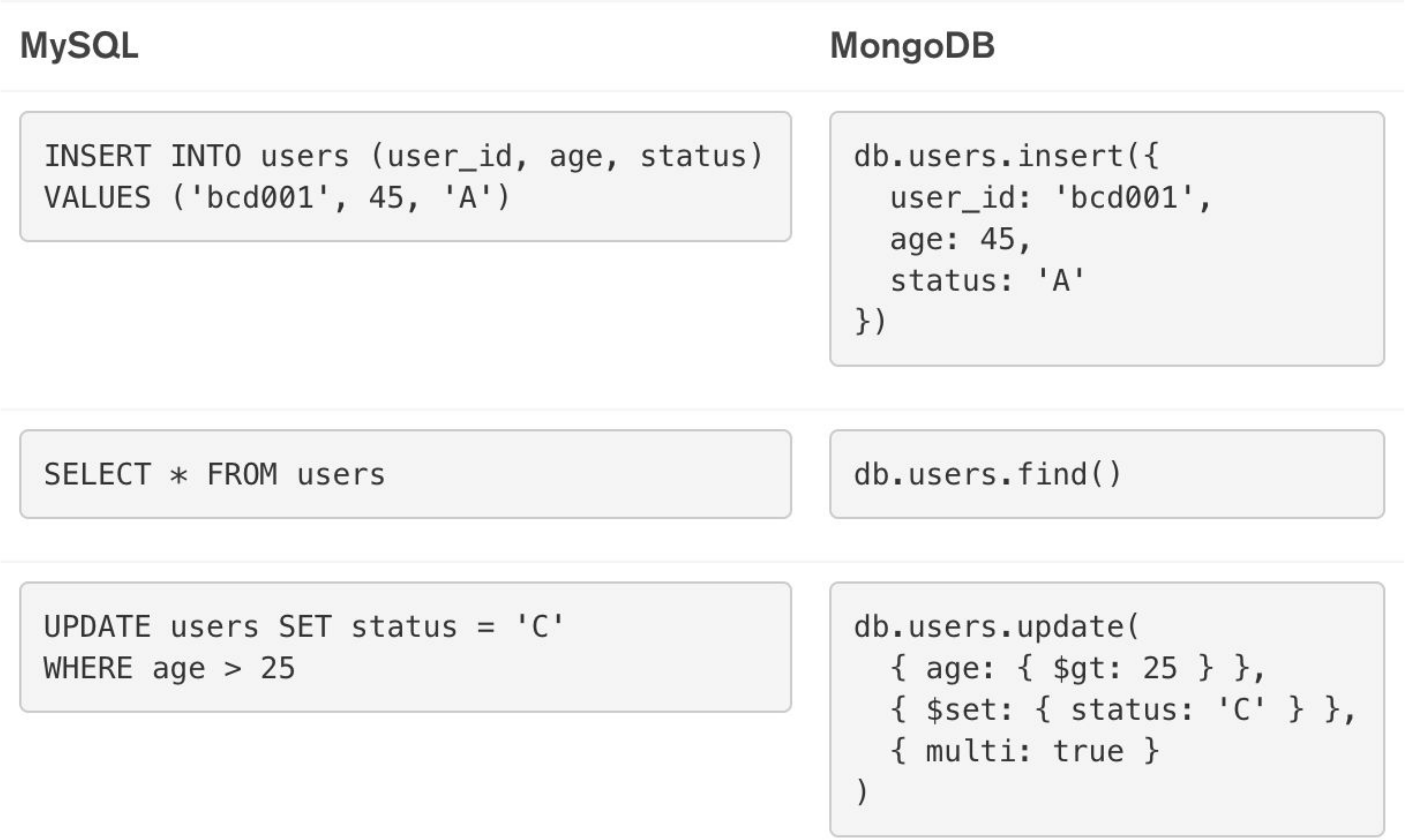
Needs to have predefined schemas to determine the structure of your data before you work with it.

All data must follow the same structure. This can require significant up-front preparation, and, as with Town A, it can mean that a change in the structure would be both difficult and disruptive to your whole system.

NoSQL


Redis
Neo4j
the graph database
Cassandra
CouchDB
relax
mongoDB
membase
riak

MongoDB Query : MySQL vs MongoDB



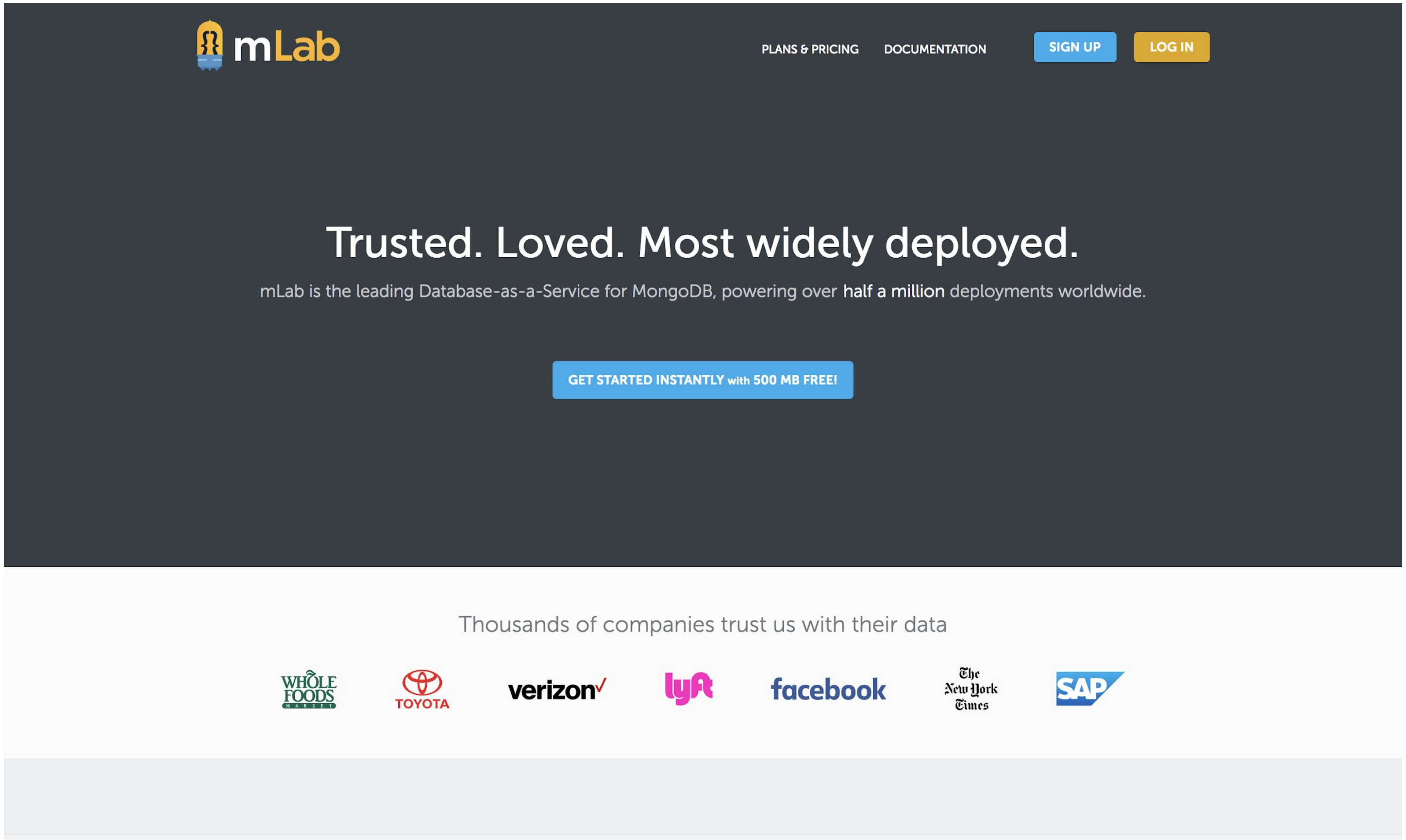
MySQL	MongoDB

`INSERT INTO users (user_id, age, status)`
`VALUES ('bcd001', 45, 'A')`
	`db.users.insert({`
	`  user_id: 'bcd001',`
	`  age: 45,`
	`  status: 'A'`
	`})`

`SELECT * FROM users`	`db.users.find()`

`UPDATE users SET status = 'C'`
`WHERE age > 25`
	`db.users.update(`
	`  { age: { $gt: 25 } },`
	`  { $set: { status: 'C' } },`
	`  { multi: true }`
	`)`

Mlab : Try and learn mongodb on mlab.com

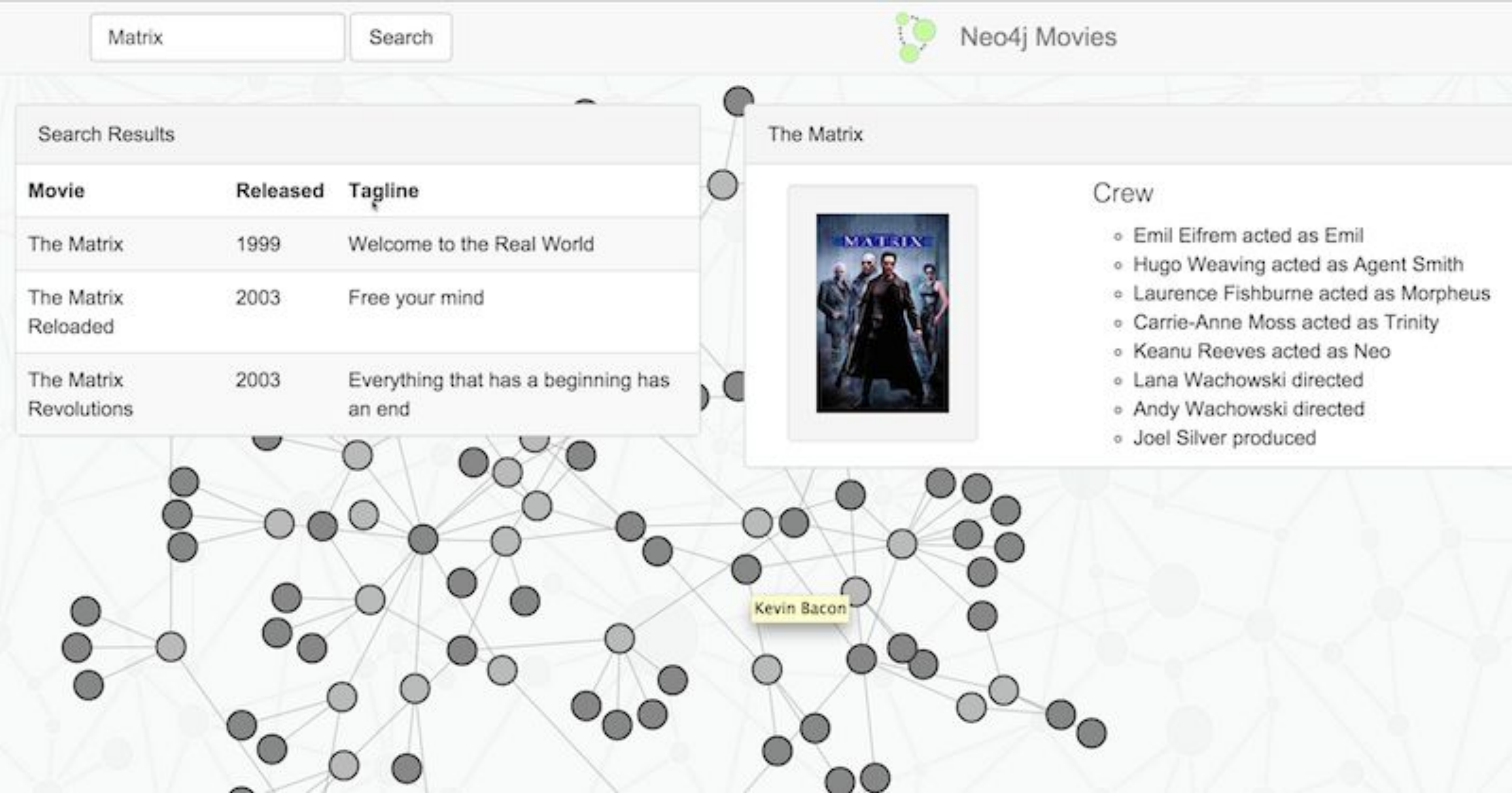


Trusted. Loved. Most widely deployed.

mLab is the leading Database-as-a-Service for MongoDB, powering over half a million deployments worldwide.

Thousands of companies trust us with their data

Graph Database



Search Results

Movie	Released	Tagline
The Matrix	1999	Welcome to the Real World
The Matrix Reloaded	2003	Free your mind
The Matrix Revolutions	2003	Everything that has a beginning has an end

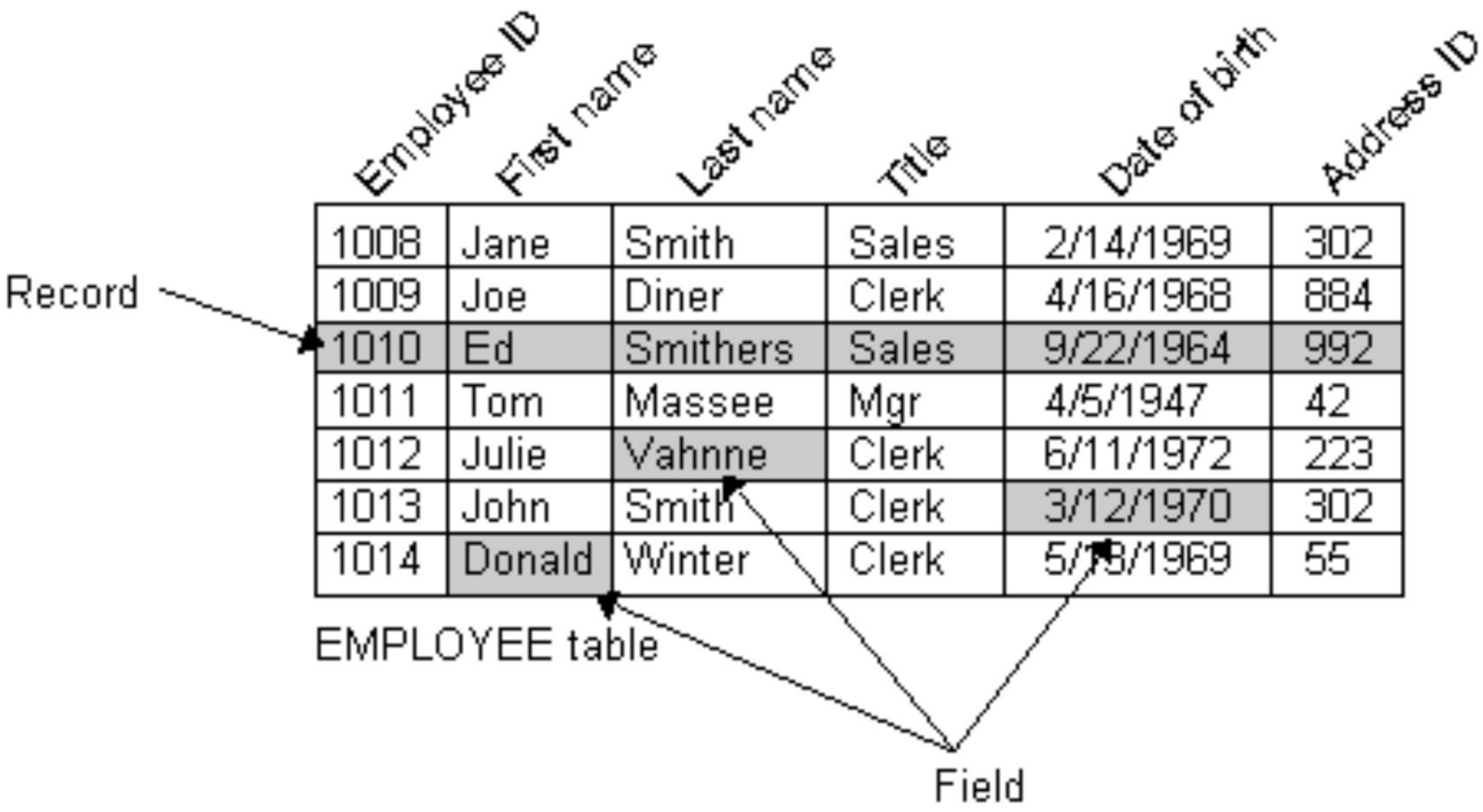
The Matrix

Crew

- Emil Eifrem acted as Emil
- Hugo Weaving acted as Agent Smith
- Laurence Fishburne acted as Morpheus
- Carrie-Anne Moss acted as Trinity
- Keanu Reeves acted as Neo
- Lana Wachowski directed
- Andy Wachowski directed
- Joel Silver produced

https://neo4j.com/developer/example-project/

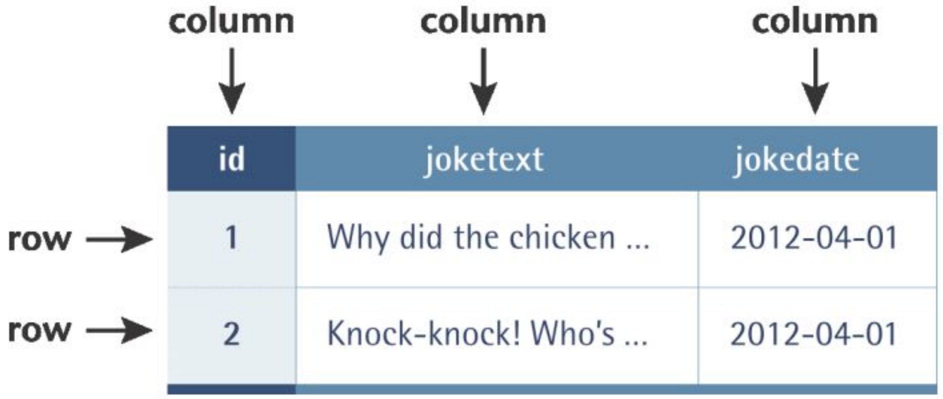
Relational Data Model



Employee ID	First name	Last name	Title	Date of birth	Address ID
1008	Jane	Smith	Sales	2/14/1969	302
1009	Joe	Diner	Clerk	4/16/1968	884
1010	Ed	Smithers	Sales	9/22/1964	992
1011	Tom	Massee	Mgr	4/5/1947	42
1012	Julie	Vahnne	Clerk	6/11/1972	223
1013	John	Smith	Clerk	3/12/1970	302
1014	Donald	Winter	Clerk	5/18/1969	55

INTRODUCTION TO MYSQL

- A database server is a program that can store large amounts of information in an organized format
- It is easily accessible through programming, eg via PHP
- Database is composed of:
  - One or more tables
  - Each table has one or more columns and rows.



id	joketext	jokedate
1	Why did the chicken ...	2012-04-01
2	Knock-knock! Who's ...	2012-04-01

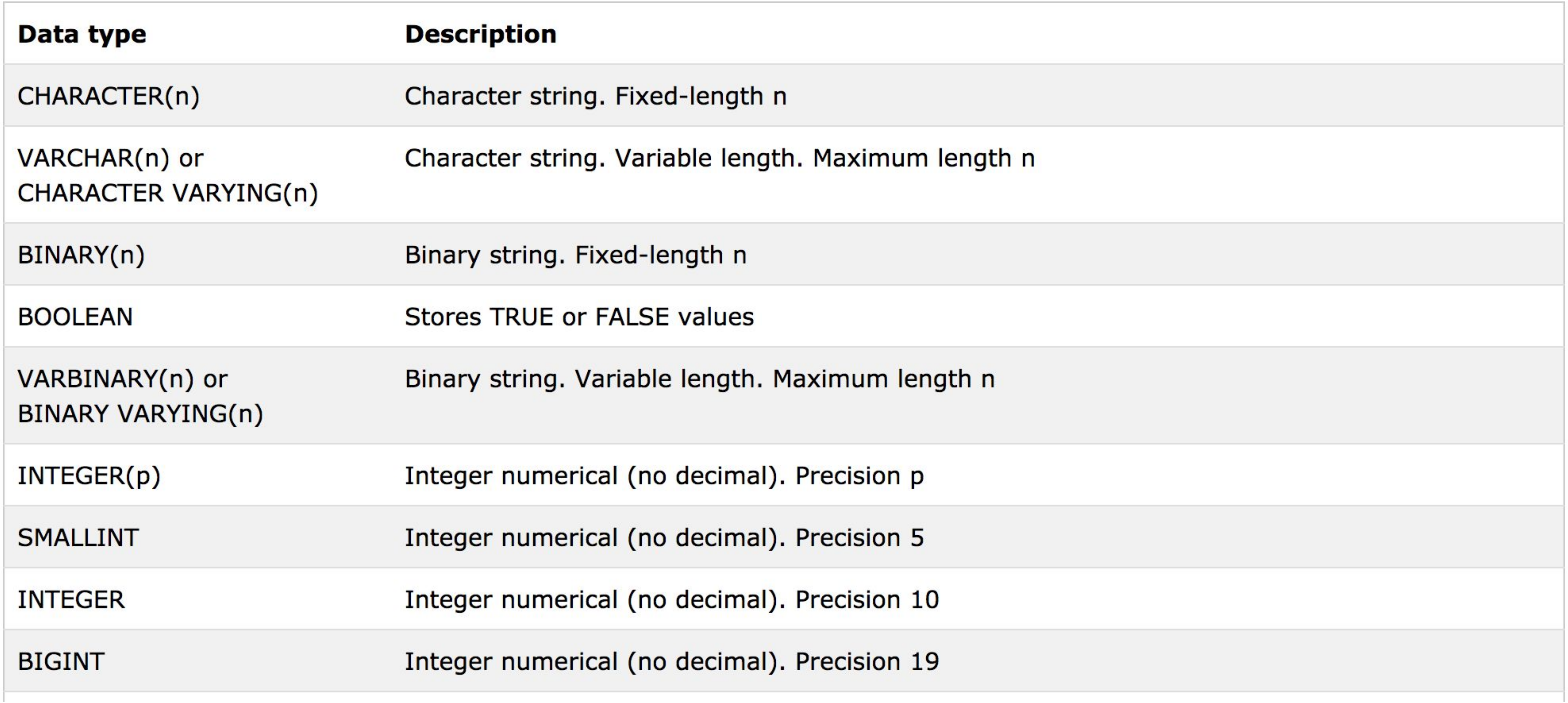
Create database

- SHOW DATABASES;
- DROP DATABASE test;
- CREATE DATABASE ijdb;

STRUCTURED COMMAND LANGUAGE (SQL)

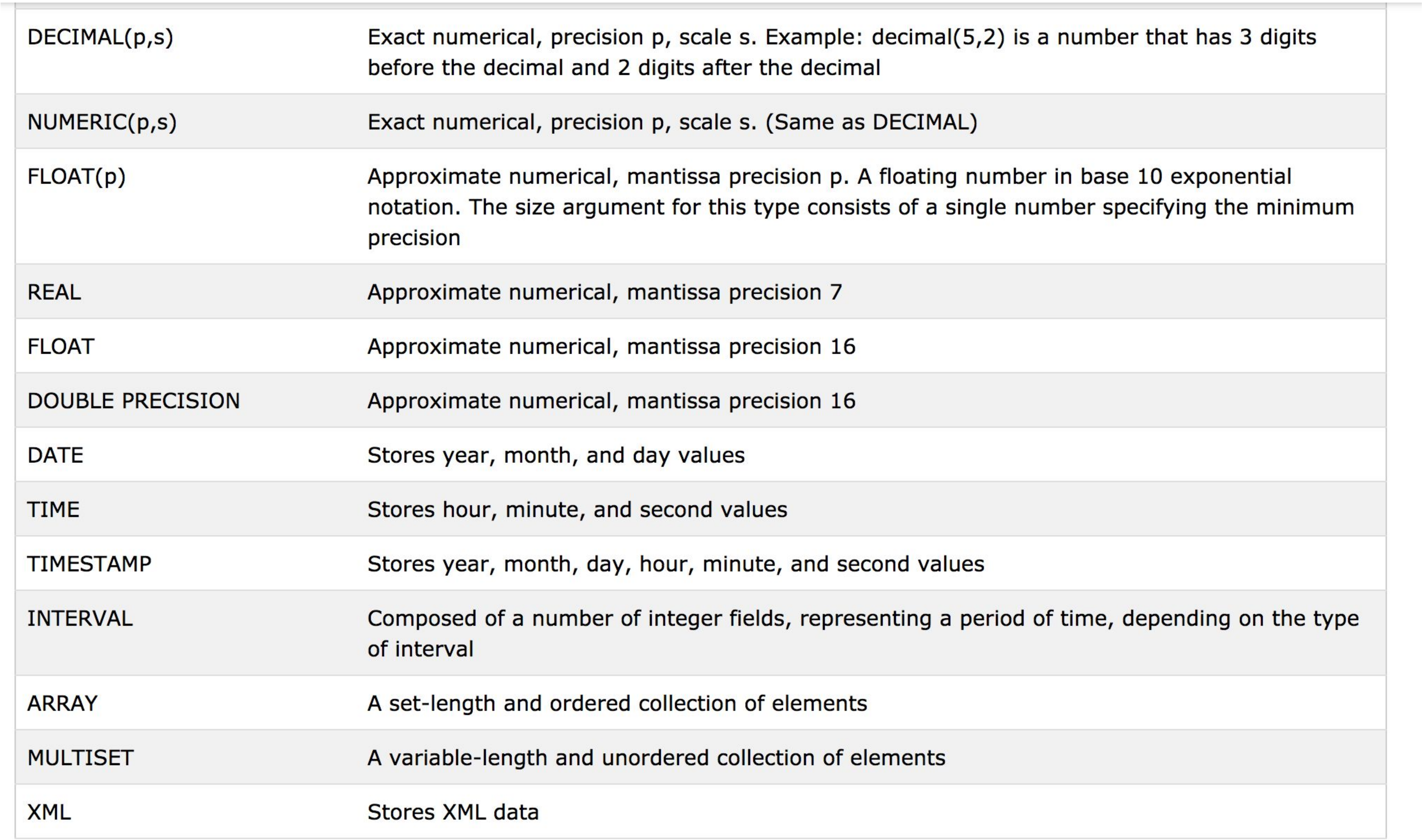
- Standard language for interacting with most databases
- SQL command is not case sensitive
- SQL is the language that let you access and manipulate database.

SQL Datatype



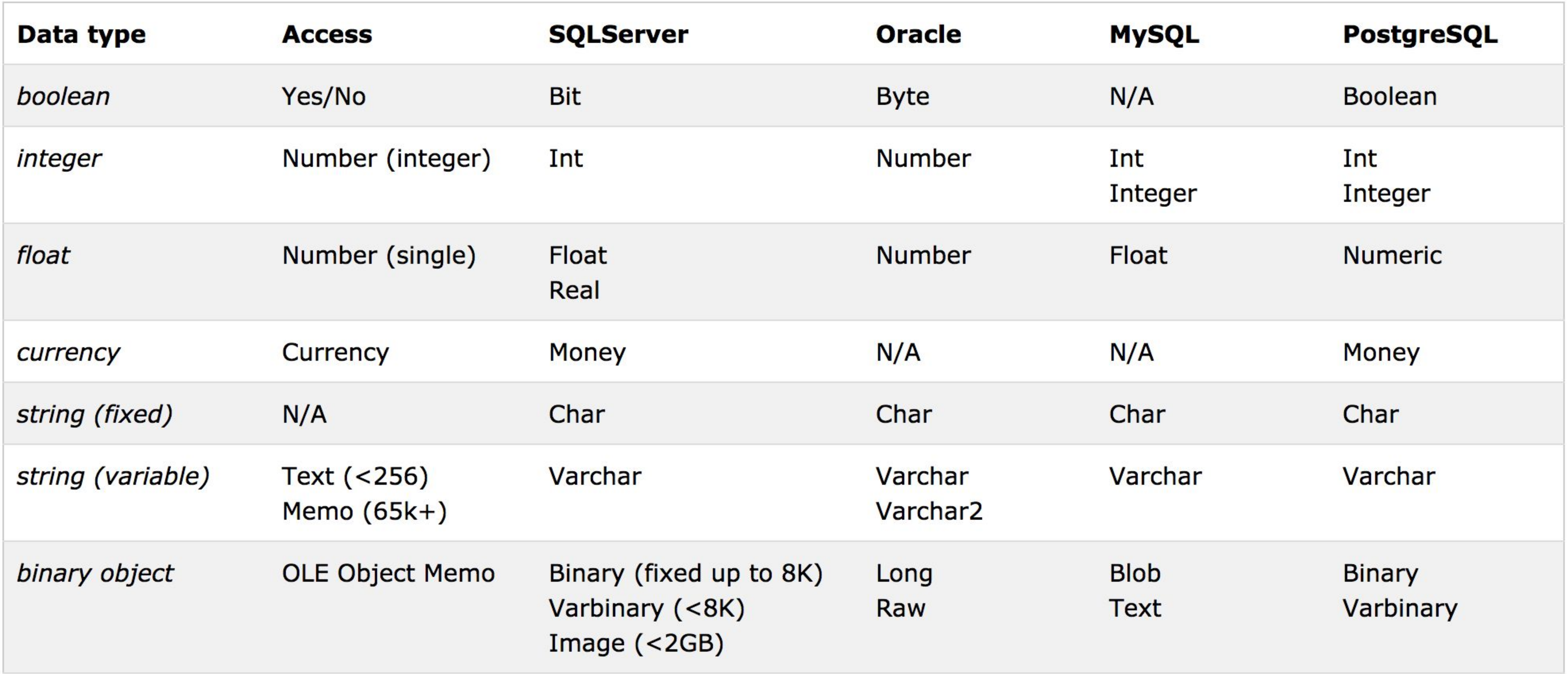
Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19

SQL Datatype (2)



DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC(p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)
FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision
REAL	Approximate numerical, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

SQL Datatype



Data type	Access	SQLServer	Oracle	MySQL	PostgreSQL
boolean	Yes/No	Bit	Byte	N/A	Boolean
integer	Number (integer)	Int	Number	Int Integer	Int Integer
float	Number (single)	Float Real	Number	Float	Numeric
currency	Currency	Money	N/A	N/A	Money
string (fixed)	N/A	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
binary object	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

Creating New Table

```
CREATE TABLE table_name (
column1Name column1Type column1Details,
column2Name column2Type column2Details,
⋮
) DEFAULT CHARACTER SET charset ENGINE=InnoDB
```

Example

```
CREATE TABLE jokes (
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
joketext TEXT,
jokedate DATE NOT NULL
) DEFAULT CHARACTER SET utf8 ENGINE=InnoDB
```

Show and describe tables.

- SHOW TABLES;
- DESCRIBE joke;
- DROP TABLE *tablename;*

Insert into database.

```
INSERT INTO tableName
(column1Name, column2Name, …)
VALUES (column1Value, column2Value, …)
```

Example:

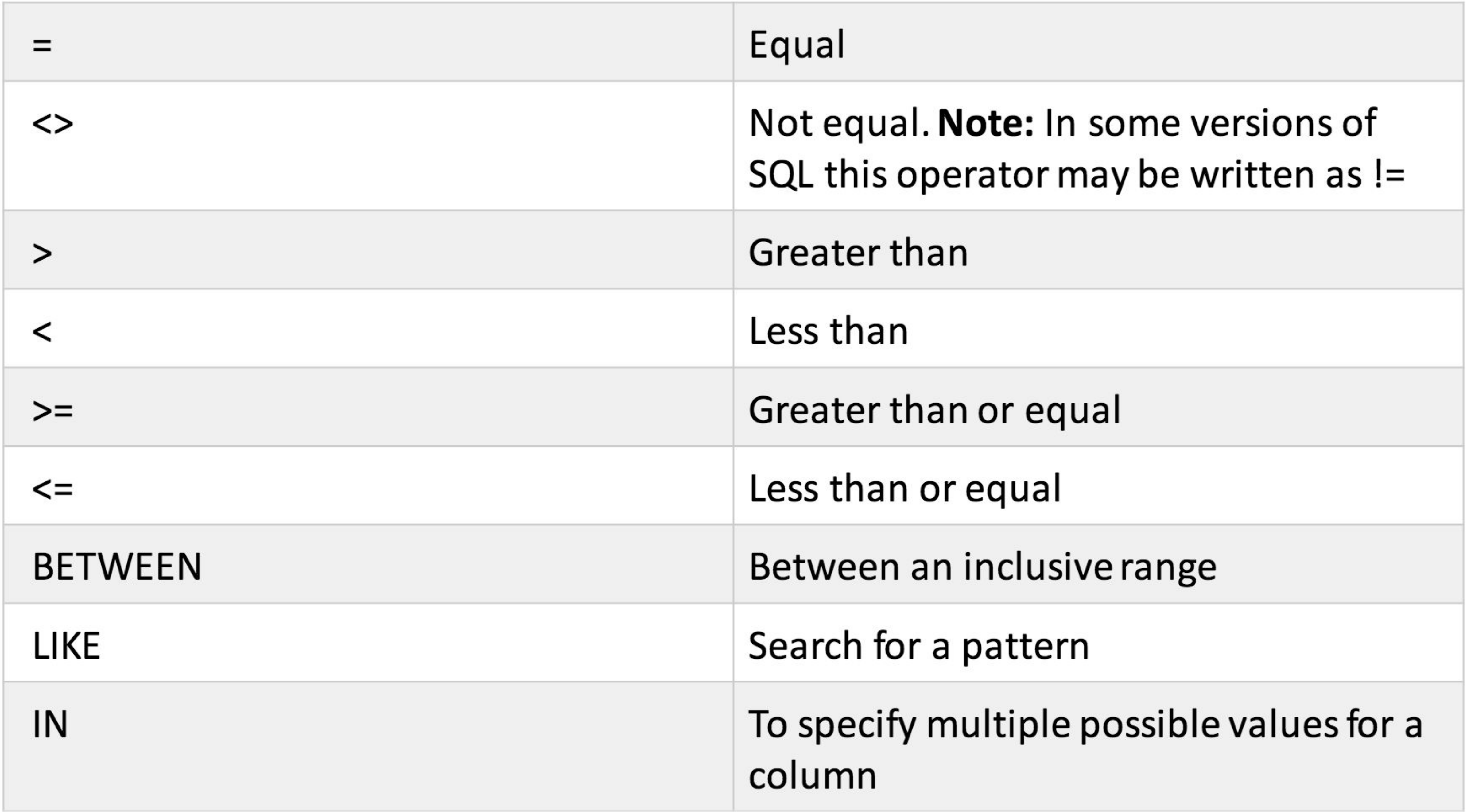
```
INSERT INTO joke
(joketext, jokedate) VALUES (
"Why did the chicken cross the road? To get to the other side!",
"2012-04-01")
```

VIEWING STORED DATA

```
SELECT * FROM joke
SELECT id, jokedate FROM joke
SELECT id, LEFT(joketext, 20), jokedate FROM joke
SELECT COUNT(*) FROM joke
SELECT COUNT(*) FROM joke WHERE jokedate >= “2012-01-01"
SELECT joketext FROM joke WHERE joketext LIKE “%chicken%"

SELECT joketext FROM joke WHERE
joketext LIKE "%knock%" AND
jokedate >= "2012-04-01" AND
jokedate < “2012-05-01"
```

MySQL Operators



=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

MODIFYING STORED DATA

```
UPDATE tableName SET
colName = newValue, …
WHERE conditions
```

Example:

```
UPDATE joke SET jokedate = "2013-04-01" WHERE id = “1"
```

```
UPDATE joke SET jokedate = "2010-04-01"
WHERE joketext LIKE "%chicken%"
```

DELETING STORED DATA

```
DELETE FROM tableName WHERE conditions
```

```
DELETE FROM joke WHERE joketext LIKE "%chicken%"
```

ALTER TABLE

```
ALTER TABLE joke ADD COLUMN authorname VARCHAR(255)

ALTER TABLE joke DROP COLUMN authorname
```

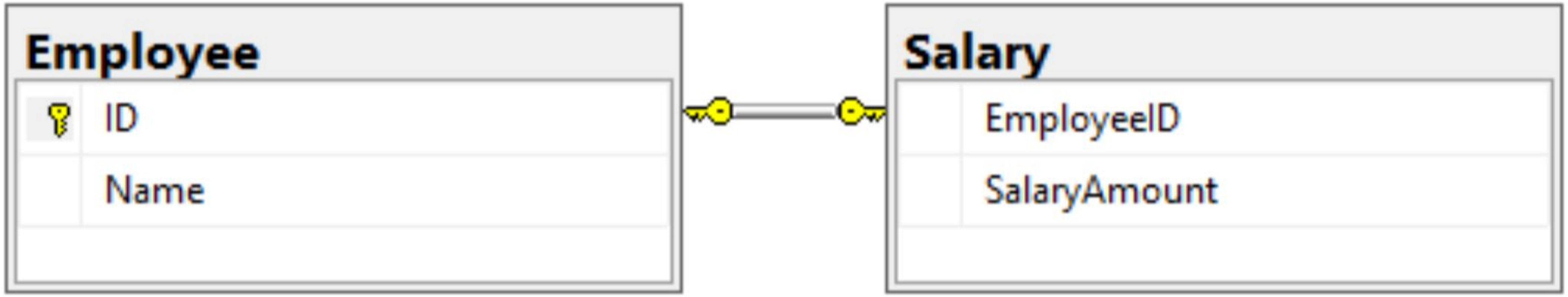
JOINING TABLE

```
SELECT joke.id, LEFT(joketext, 20), name, email
FROM joke INNER JOIN author
ON authorid = author.id
```

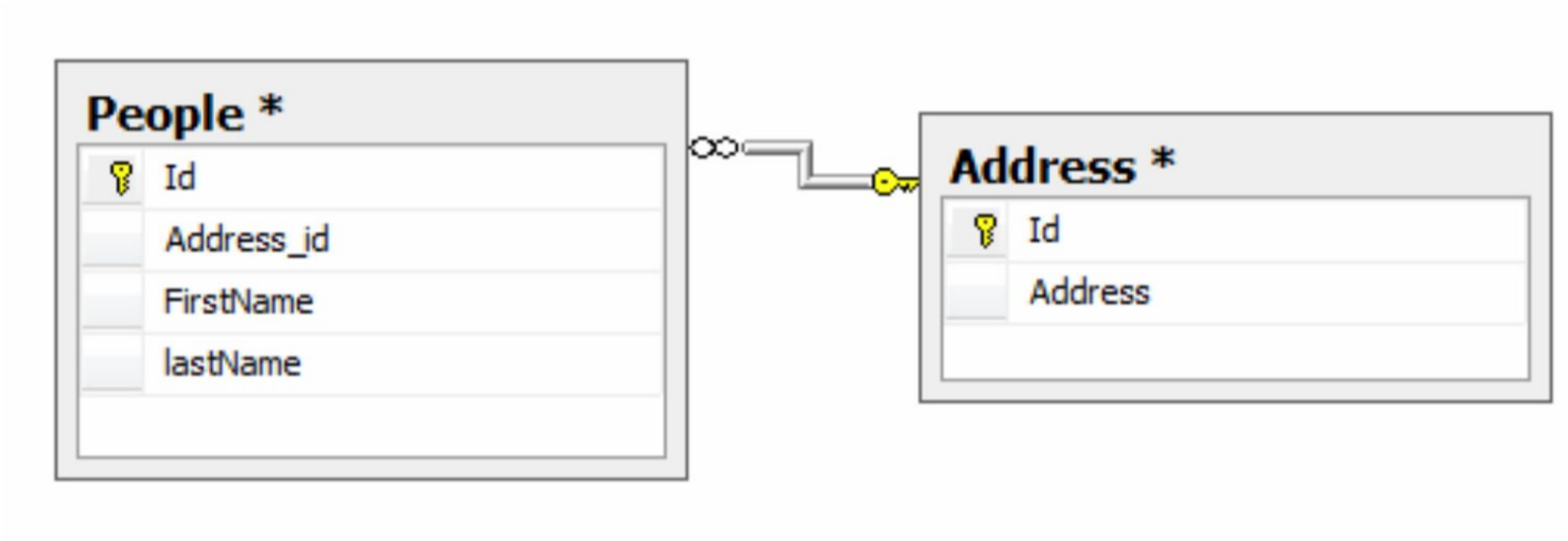
RELATIONSHIPS

- One-to-One Relationships
- One to Many Relationships
- Many-to-Many Relationships

One to One Relationship



One to Many Relationship



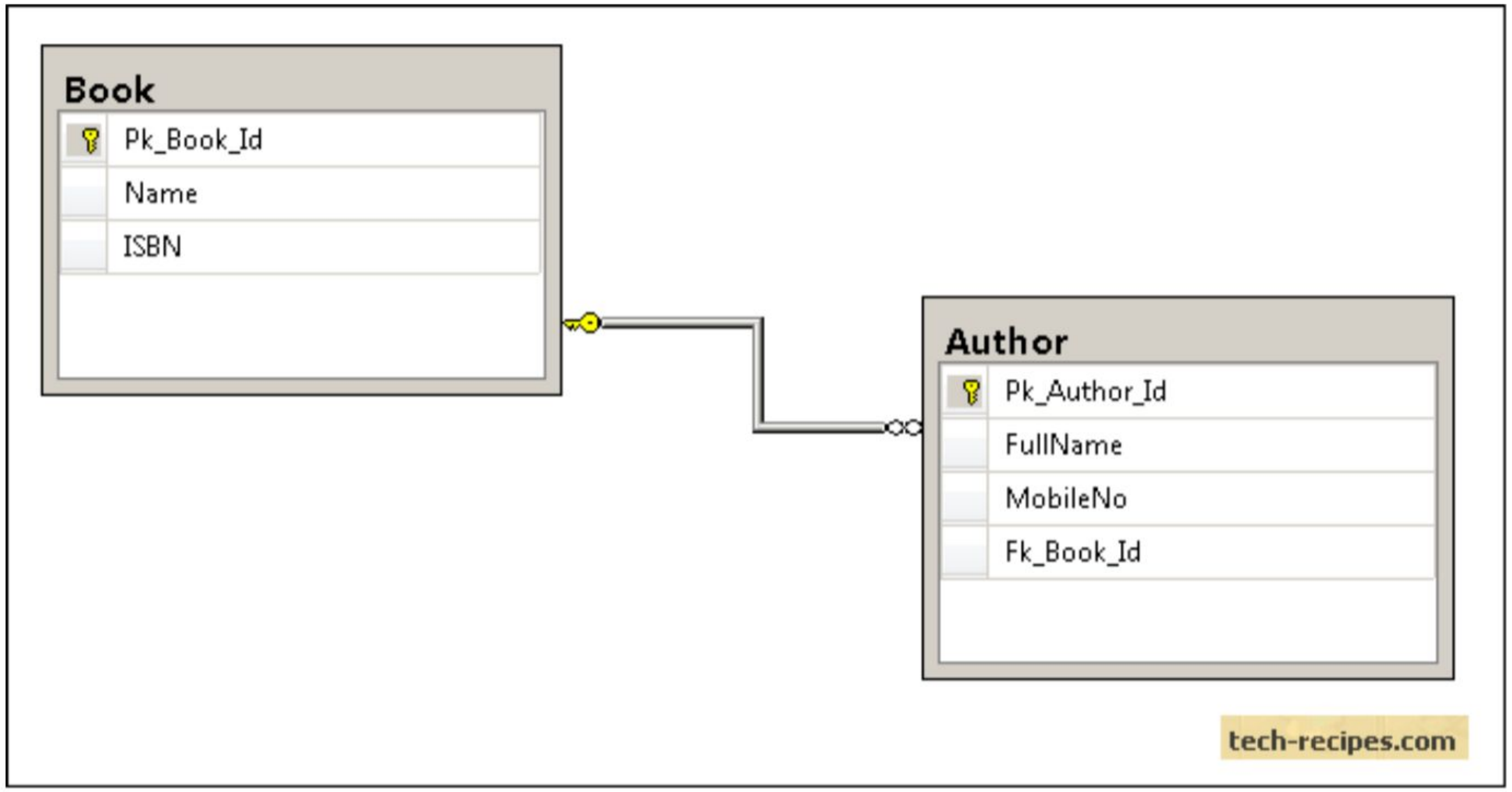
People *

- Id
- Address_id
- FirstName
- lastName

Address *

- Id
- Address

Many to Many Relationship



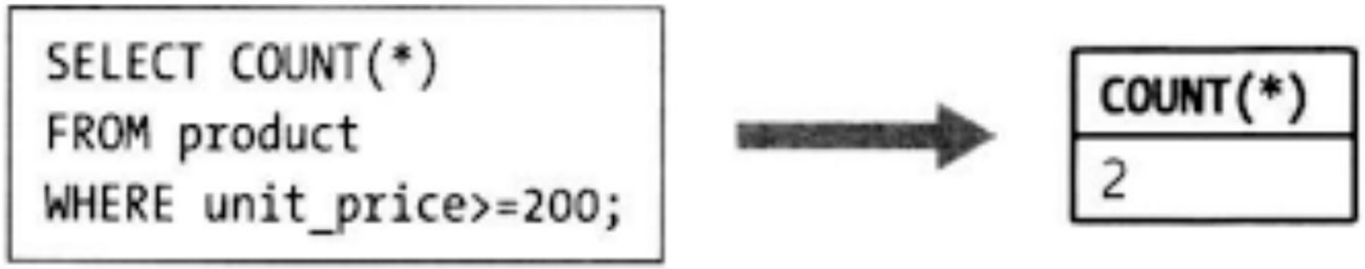
SORTING QUERY

```
SELECT id, name, email FROM author ORDER BY name
```

```
SELECT id, name, email FROM author ORDER BY name DESC
```

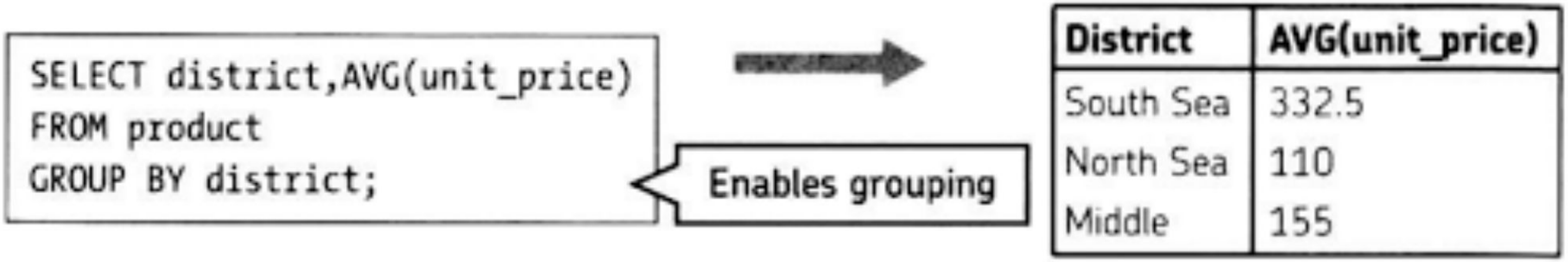
```
SELECT joke.id, joketext, name, email
FROM joke INNER JOIN author
ON authorid = author.id
ORDER BY timesviewed DESC
LIMIT 10
```

MySQL Aggregation



```
SELECT COUNT(*)
FROM product
WHERE unit_price>=200;
```

COUNT(*)
2



```
SELECT district,AVG(unit_price)
FROM product
GROUP BY district;
```

District	AVG(unit_price)
South Sea	332.5
North Sea	110
Middle	155

HAVING

```
SELECT author.name, COUNT(joke.id) AS numjokes
FROM author LEFT JOIN joke
ON authorid = author.id
WHERE numjokes = 0
GROUP BY author.id
```