Intro to Backend Development

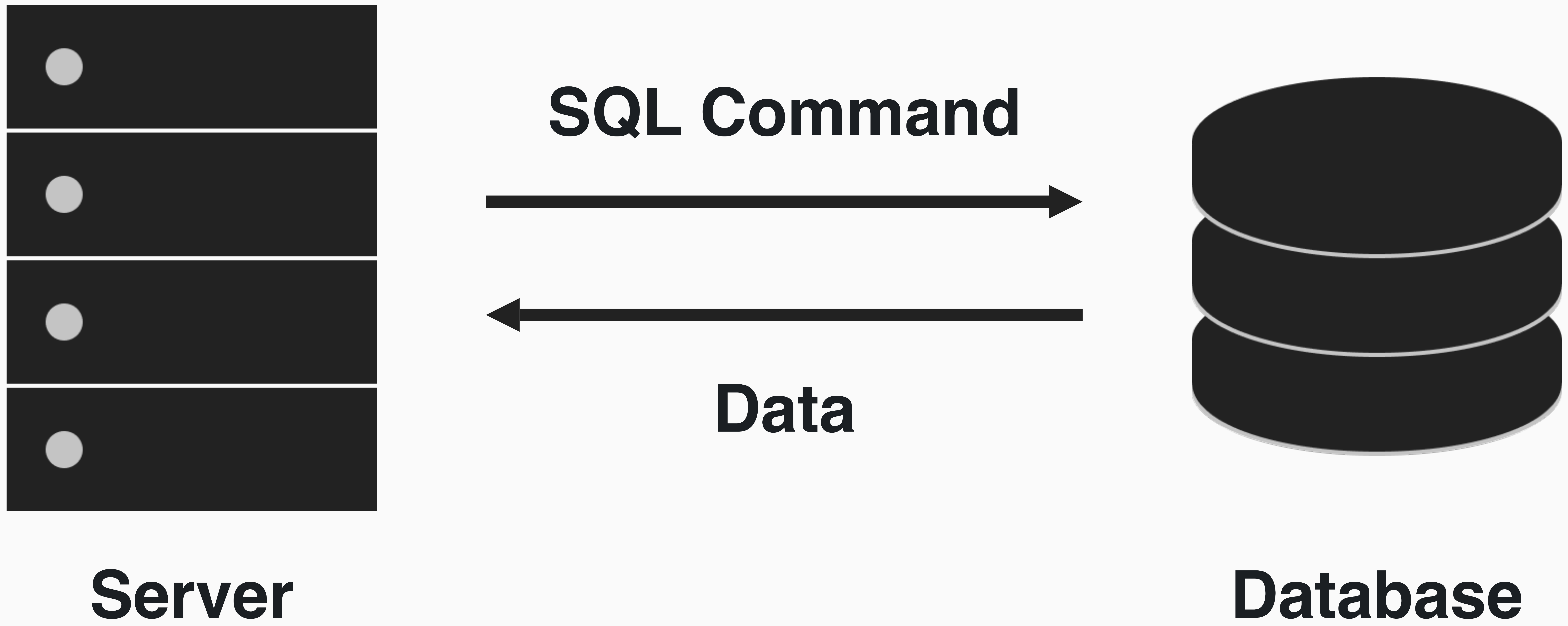
Lecture 4 · Abstractions

Kidus Zegeye
Tony Matchev

Announcements

- PA3 due Wednesday
- Remember to double-check route spellings!
- Always test in Postman before running the test script
- Follow file submission format in assignment handout EXACTLY

Review


SQL Command
Data
Server
Database

Relationships

- Relate data to other data in a structured manner
- Use Foreign Key columns to create relationships
- One-to-One
- One-to-Many
- Many-to-Many

User Table

id	name
10	“Jessica”
20	“Shungo”

Profile Table

id	status	user_id
30	“online”	20
40	“away"	10

User Table

id	name
10	“Jessica”
20	“Shungo”

Profile Table

id	status	user_id
30	“online”	20
40	“away"	10

User Table

id	name
10	“Jessica”
20	“Shungo”

Profile Table

id	status	user_id
30	“online”	20
40	“away"	10

Album Table

id	name
10	“IGOR”
20	“4:44”

Song Table

id	name	album_id
30	“EARFQUAKE"	10
40	“I THINK”	10
50	“4:44”	20
60	“Moonlight”	20

Album Table

id	name
10	“IGOR”
20	“4:44”

Song Table

id	name	album_id
30	“EARFQUAKE"	10
40	“I THINK”	10
50	“4:44”	20
60	“Moonlight”	20

Album Table

id	name
10	“IGOR”
20	“4:44”

Song Table

id	name	album_id
30	“EARFQUAKE"	10
40	“I THINK”	10
50	“4:44”	20
60	“Moonlight”	20

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

Student Table

id	netid
1	“jzs27”
2	“sn685”

Join Table

student_id	course_id
1	1110
1	1998
2	1110
2	1998

Course Table

id	title
1110	“CS1110"
1998	“CS1998"

SQL Drawbacks

Painpoints

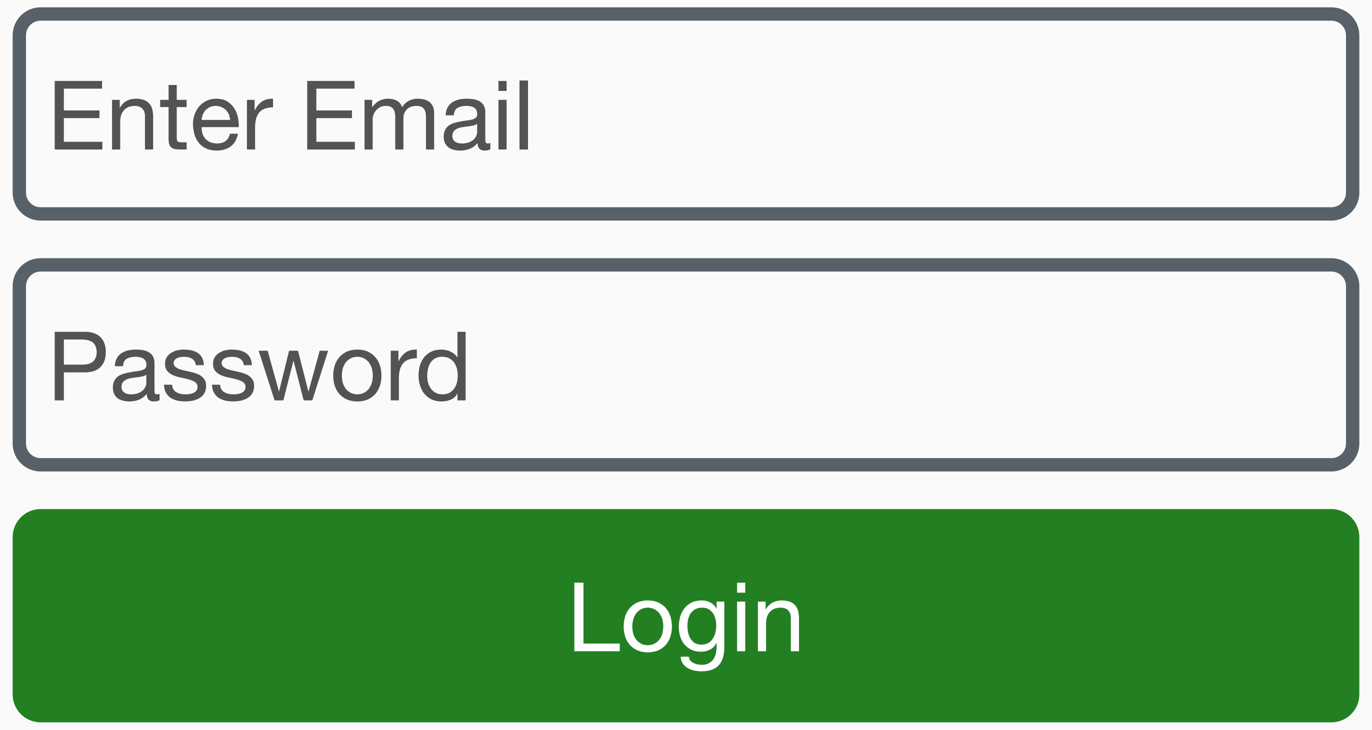
- Lower code readability
- Susceptible to vulnerabilities
- Runtime exceptions not handled well

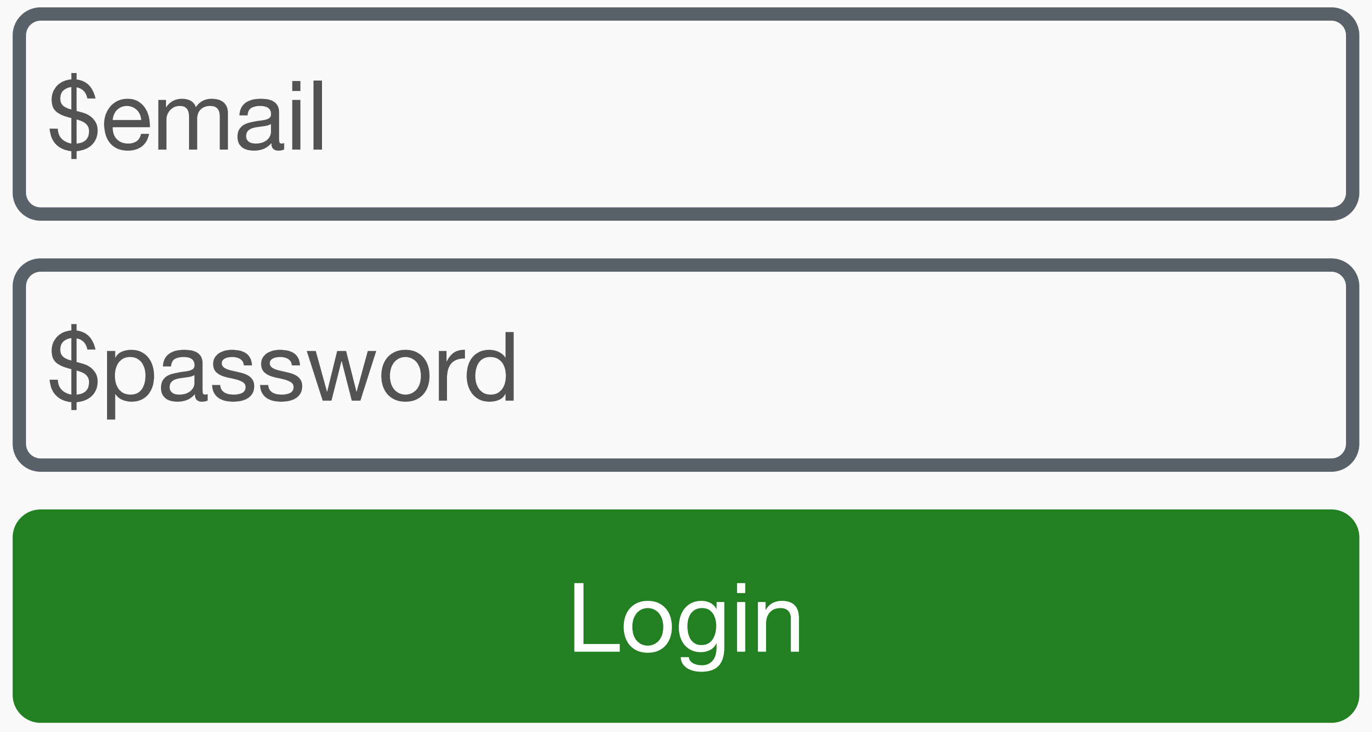
```sql
SELECT Country.name, T.language
FROM country Country,
countrylanguage CL2, (SELECT
tmp.language, code FROM country,
(SELECT CL.language,
MAX(C.surfacearea)
FROM country C, countrylanguage CL
WHERE C.code=CL.countrycode
GROUP BY CL.language) as tmp
WHERE (surfacearea = tmp.max)) T
WHERE T.code = Country.code
AND T.language = CL2.language AND
Country.code = CL2.countrycode;
```

Vulnerabilities

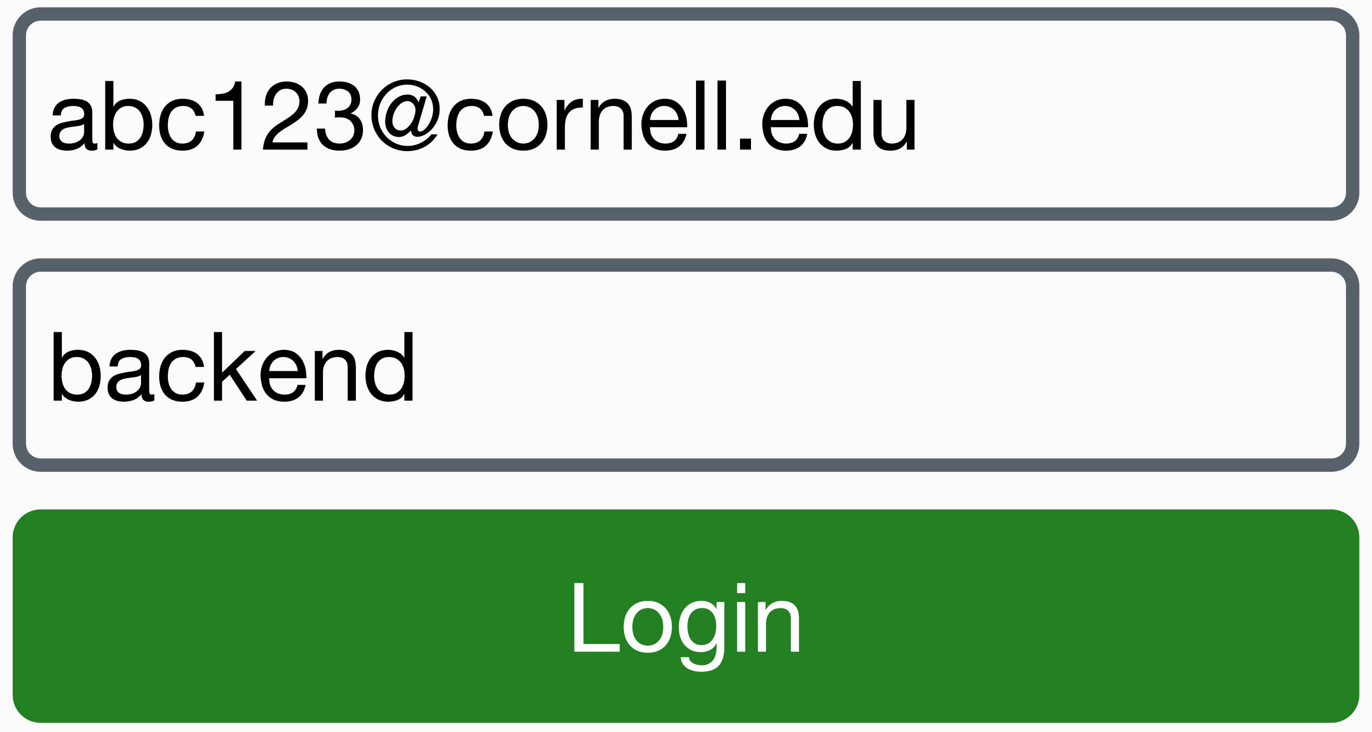
- SQL Injection: specially craft an input to trick SQL semantics
- Allow database access to a malicious external source
  - Query unauthorized data
  - Corrupt or delete existing data

Example: Breaking Login


Enter Email
Password
Login

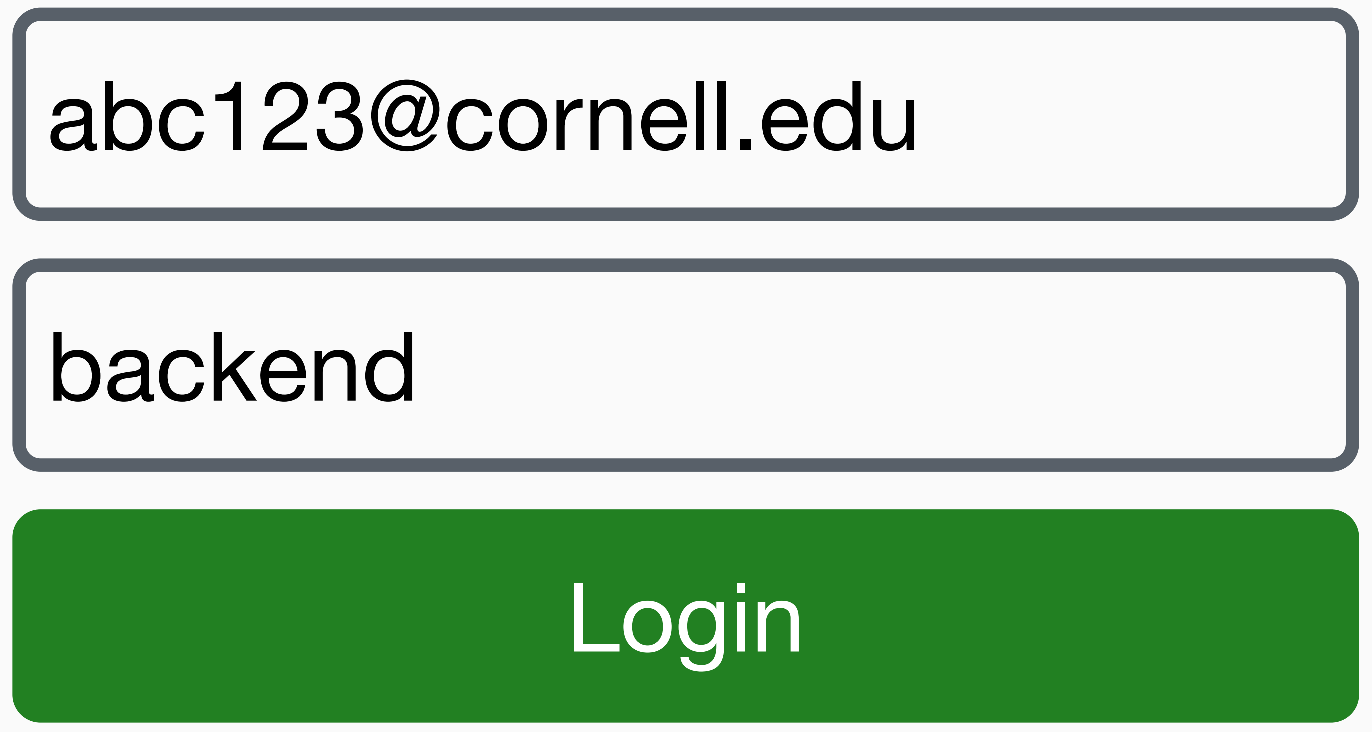


```
SELECT * FROM user WHERE
email='$email' AND password='$password';
```


abc123@cornell.edu

backend

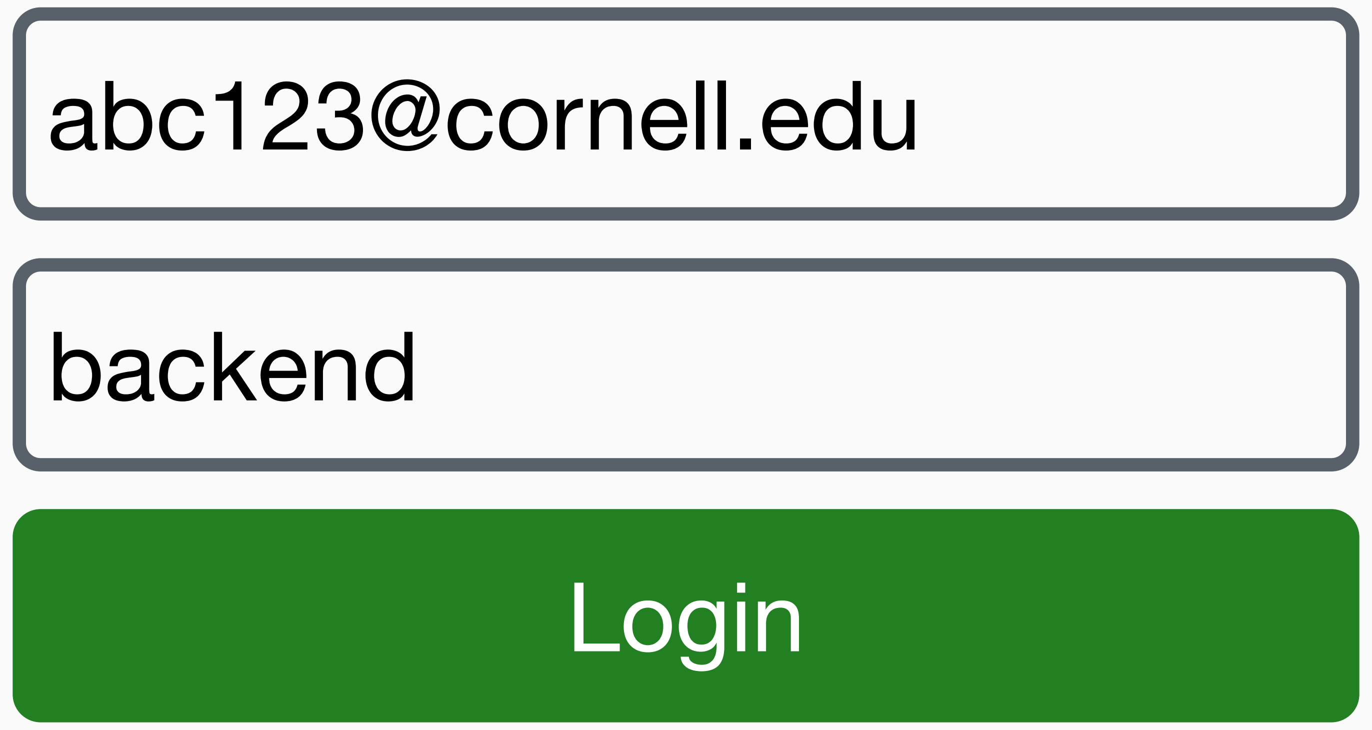
```
SELECT * FROM user WHERE
email='abc123@cornell.edu' AND password='backend';
```



abc123@cornell.edu

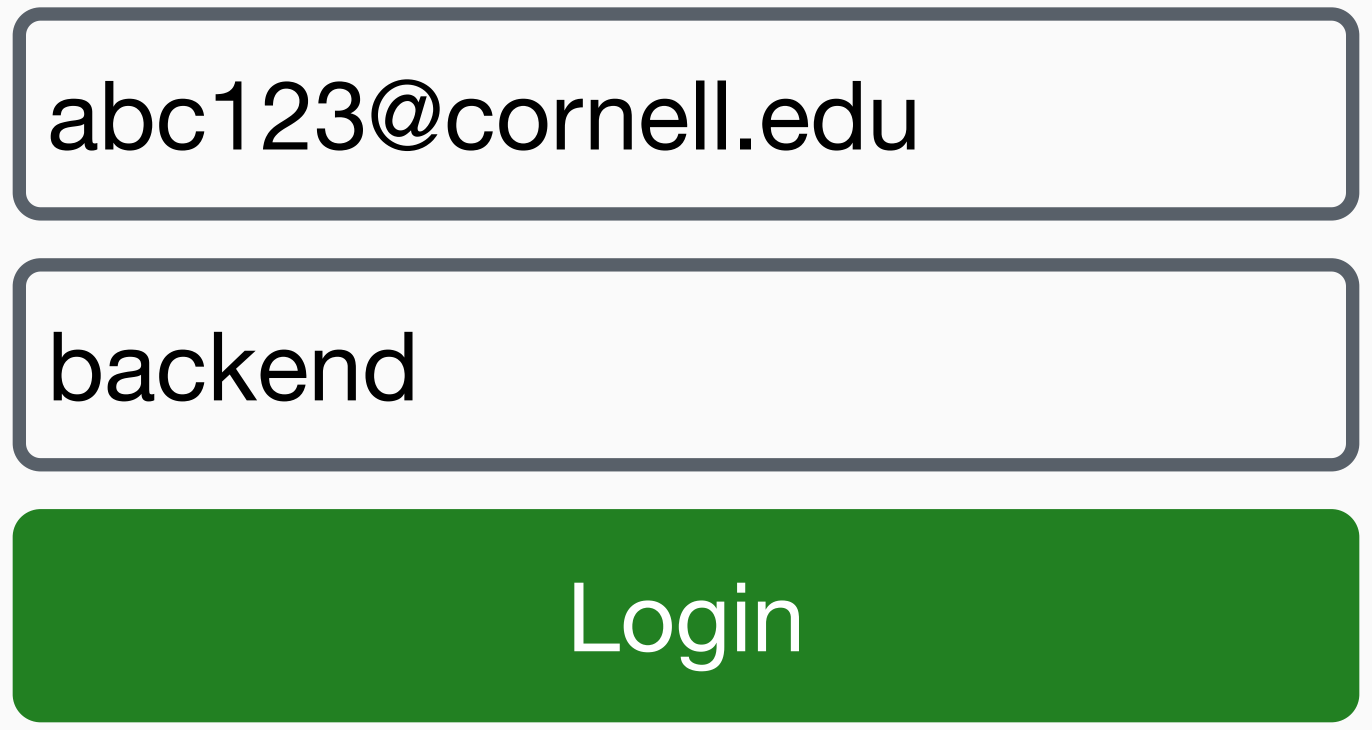
backend

```
SELECT * FROM user WHERE
FALSE AND FALSE;
```


abc123@cornell.edu

backend

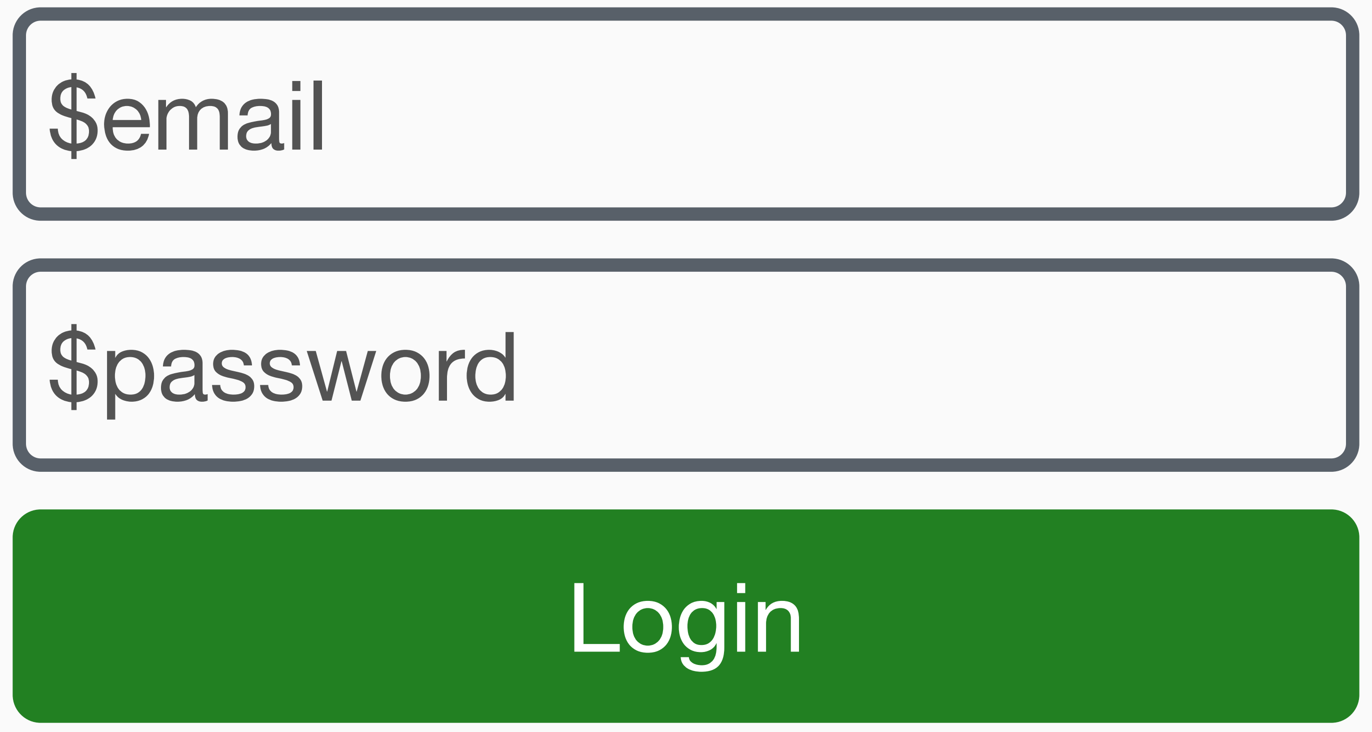
```
SELECT * FROM user WHERE
TRUE AND FALSE;
```



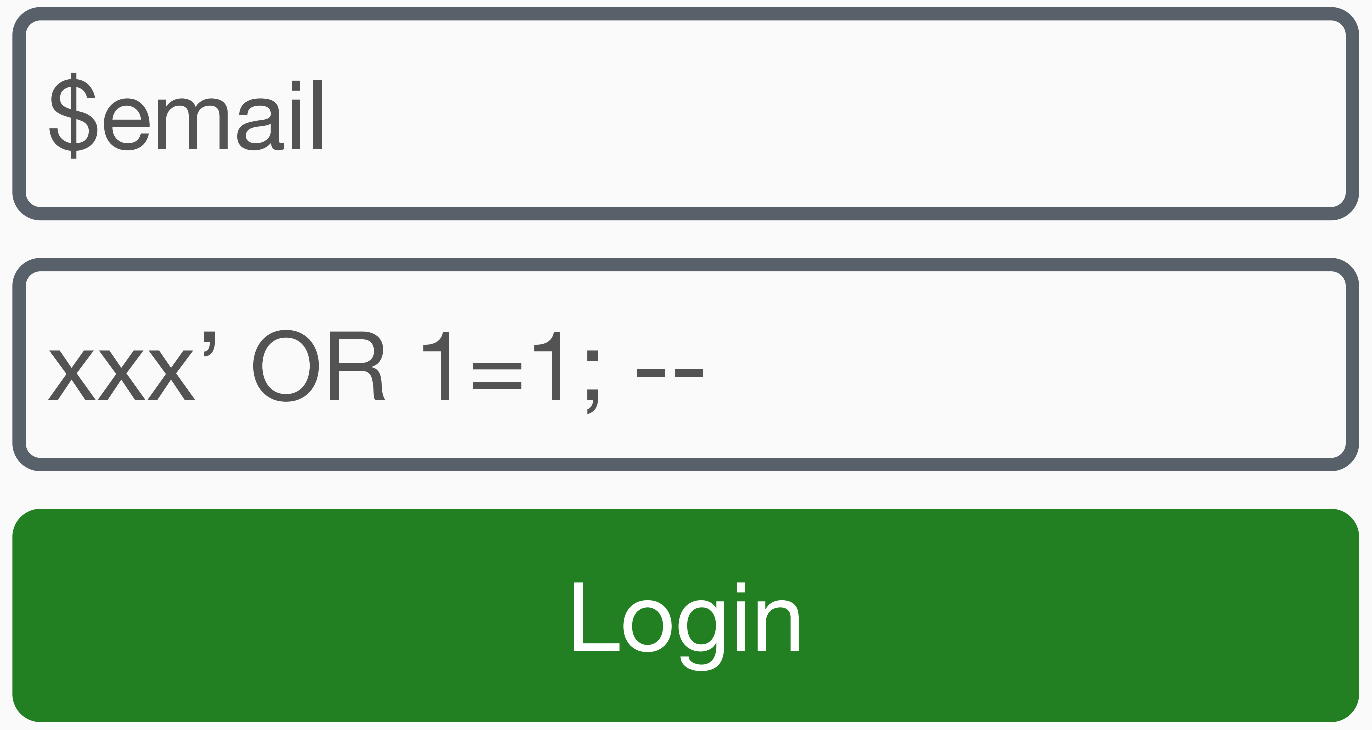
abc123@cornell.edu

backend

```
SELECT * FROM user WHERE
TRUE AND TRUE;
```

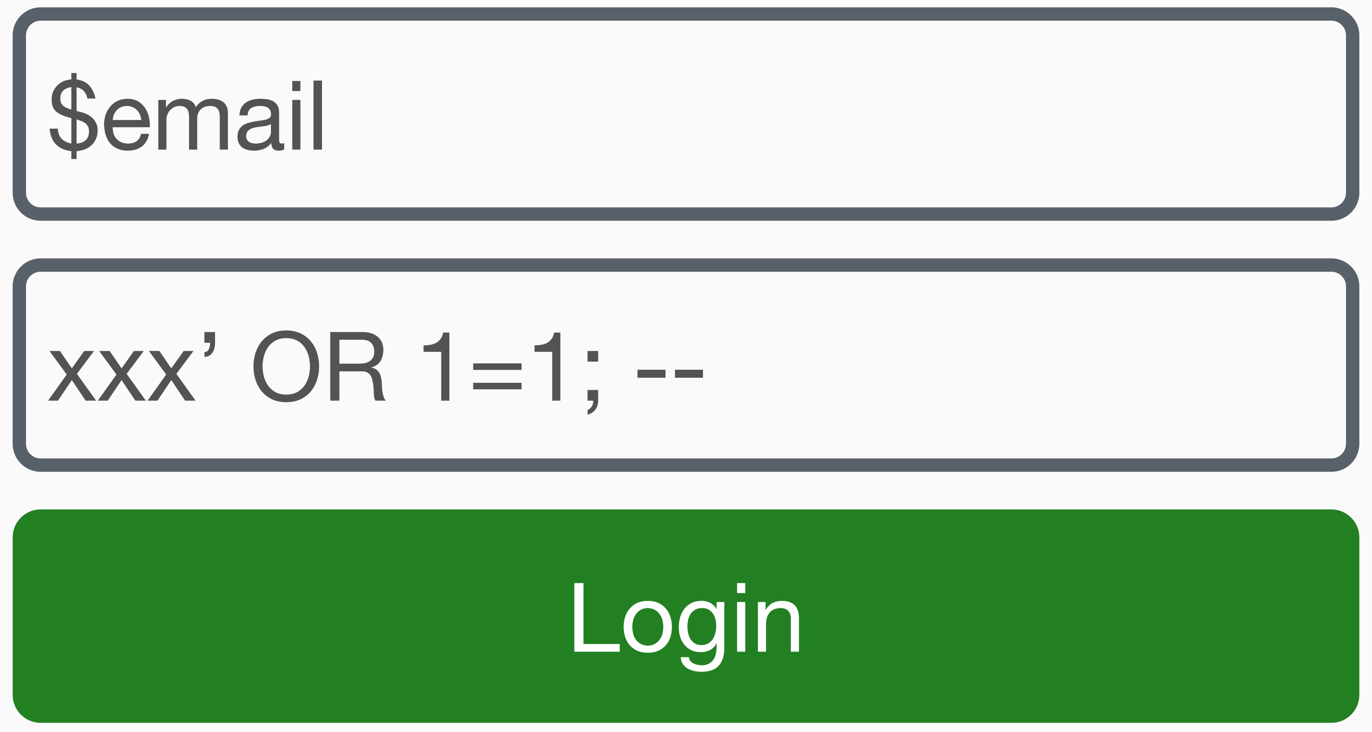


```
SELECT * FROM user WHERE
email='$email' AND password='$password';
```


$email

xxx’ OR 1=1; --

```
SELECT * FROM user WHERE
email='$email' AND password='xxx' OR 1=1; -- ';
```

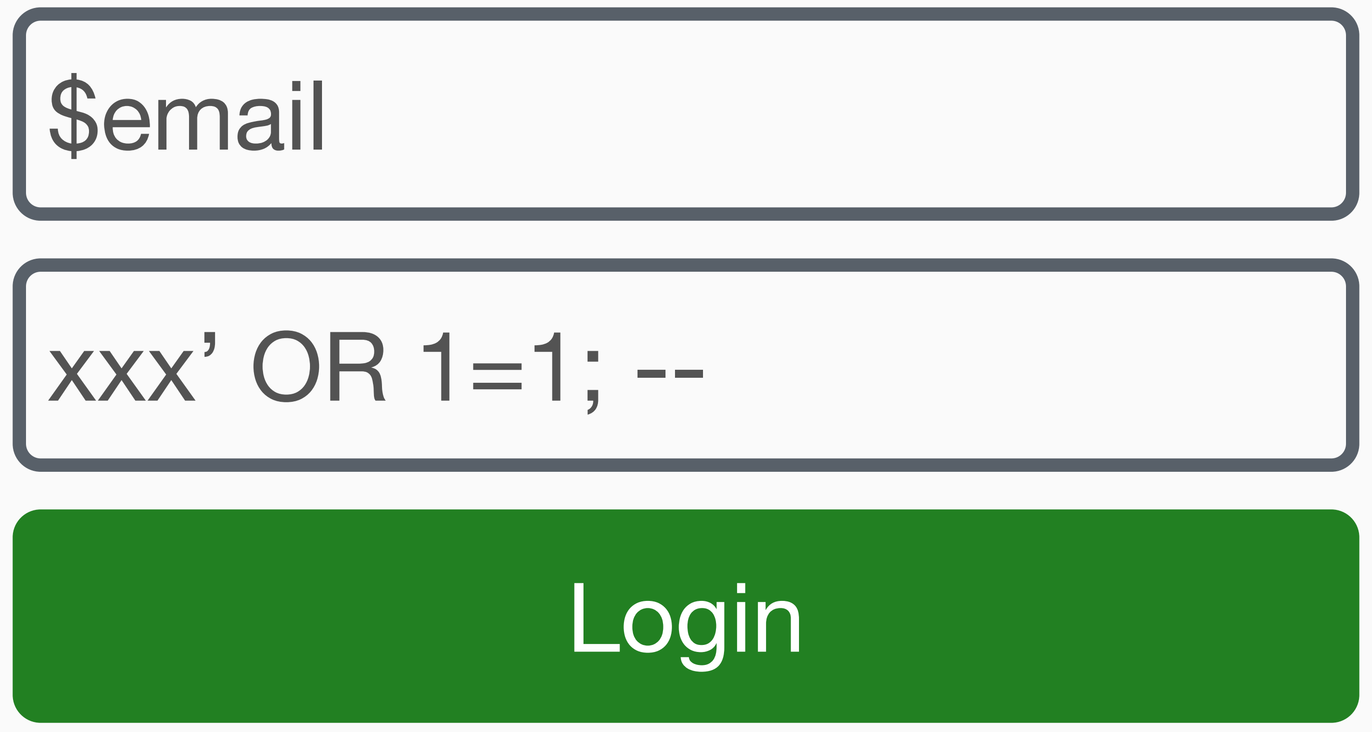

$email

xxx’ OR 1=1; --

Login

```
SELECT * FROM user WHERE
email='$email' AND password='xxx' OR 1=1; -- ';
```

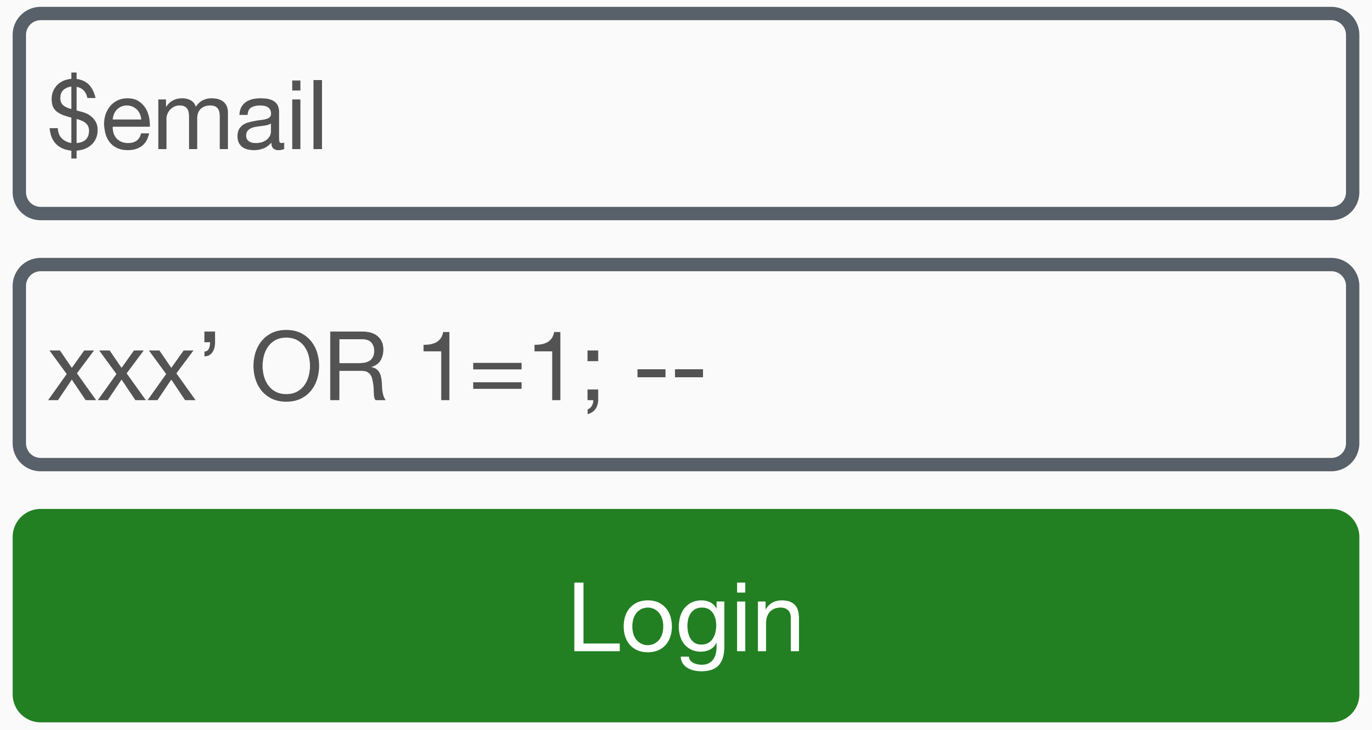
SQL Drawbacks



$email

xxx’ OR 1=1; --

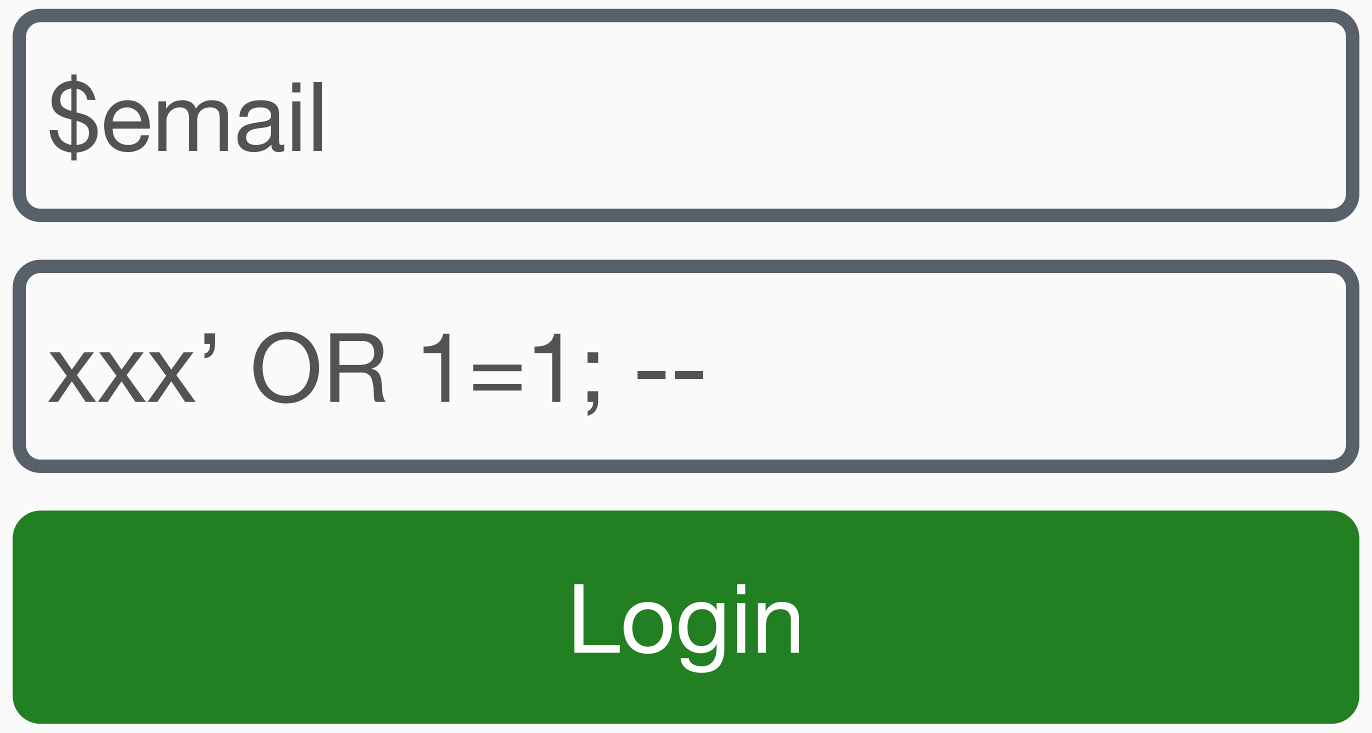
```
SELECT * FROM user WHERE
FALSE AND FALSE OR TRUE;
```



$email

xxx’ OR 1=1; --

```
SELECT * FROM user WHERE
FALSE OR TRUE;
```



$email

xxx’ OR 1=1; --

```
SELECT * FROM user WHERE
TRUE;
```

Object Relational Mapping

Objects as Abstractions

- Query and manipulate data using **objects**
- Still preserve the database for SQL and storage benefits
- Allows us to reason about entries as instances of an object

```
CREATE TABLE user (
  id   INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  age  INTEGER NOT NULL
);
```

SQL

```
class User:
  id = Column(
         Integer,
         primary_key=True
       )
  name = Column(String)
  age = Column(Integer)
```

ORM

SQLAlchemy

- Popular ORM for python
- Comes with built in query patterns
- Integrates seamlessly with Flask

ORM Benefits

- Readable code that more closely resembles our thoughts
- Abstract, SQL-free data representation
- Database and application use same logic
- Cross-compatibility with other databases
- Secure against SQL vulnerabilities (i.e. Injection)

ORM Drawbacks

- Slower query performance for more complex queries
- Increases application code (SQL + abstraction code)
- Abstracts away fundamentals of SQL
- Potential for misuse because too abstract

app.py vs db.py

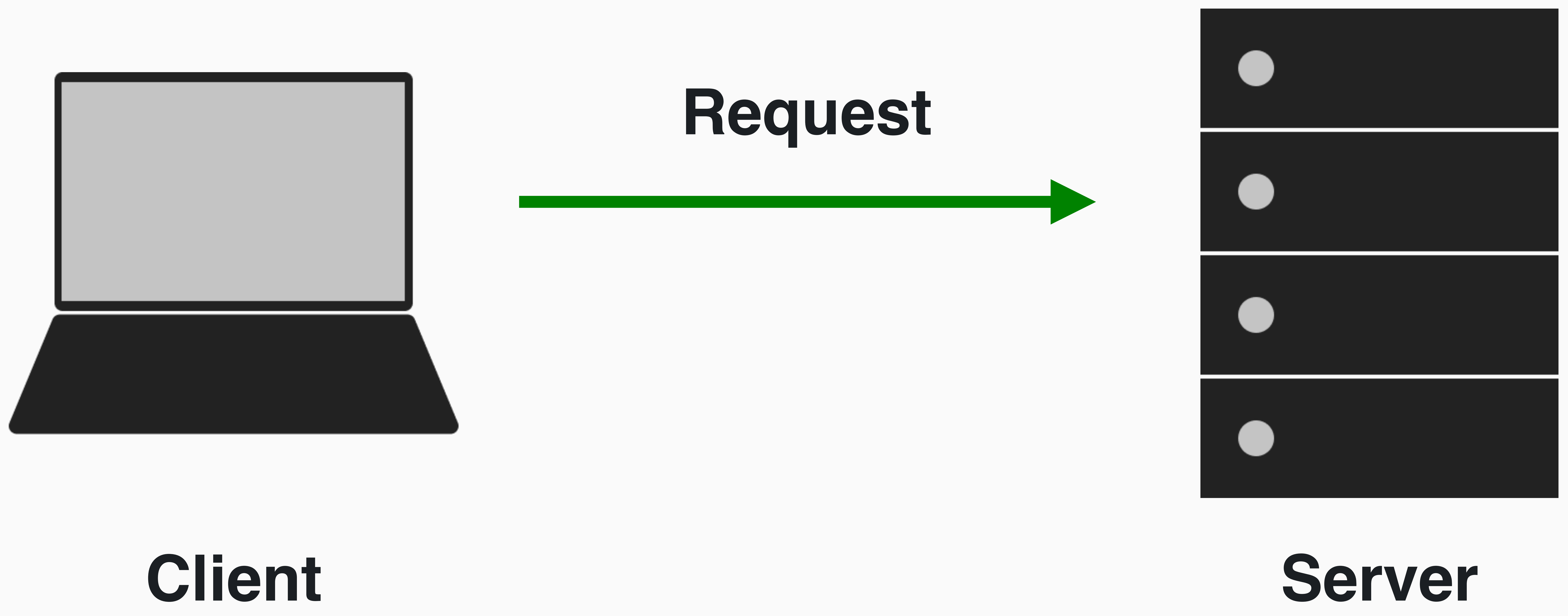
app.py

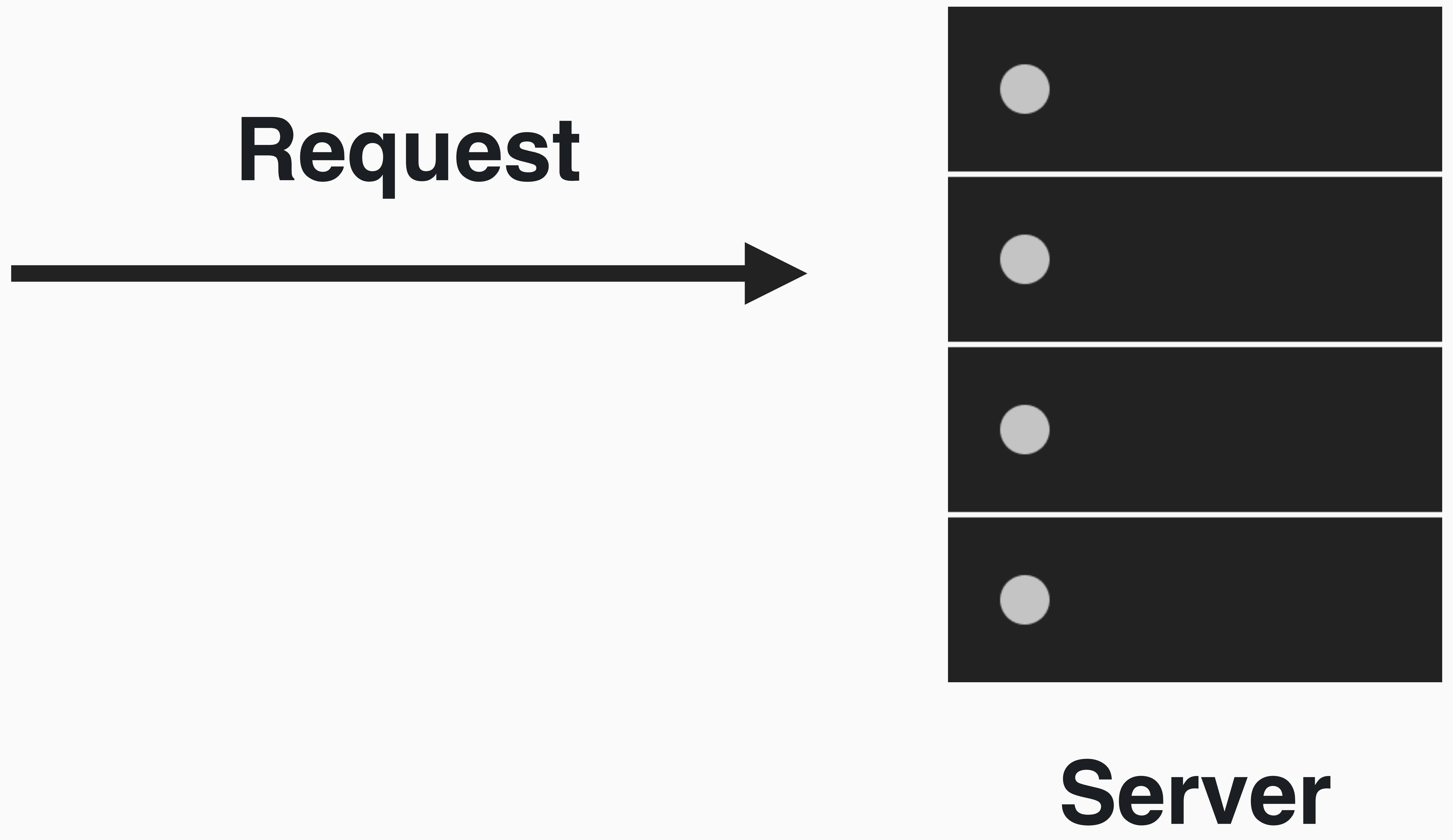
- Connects server to client
- Defines routes and responses
- Handles all logic and operations

db.py

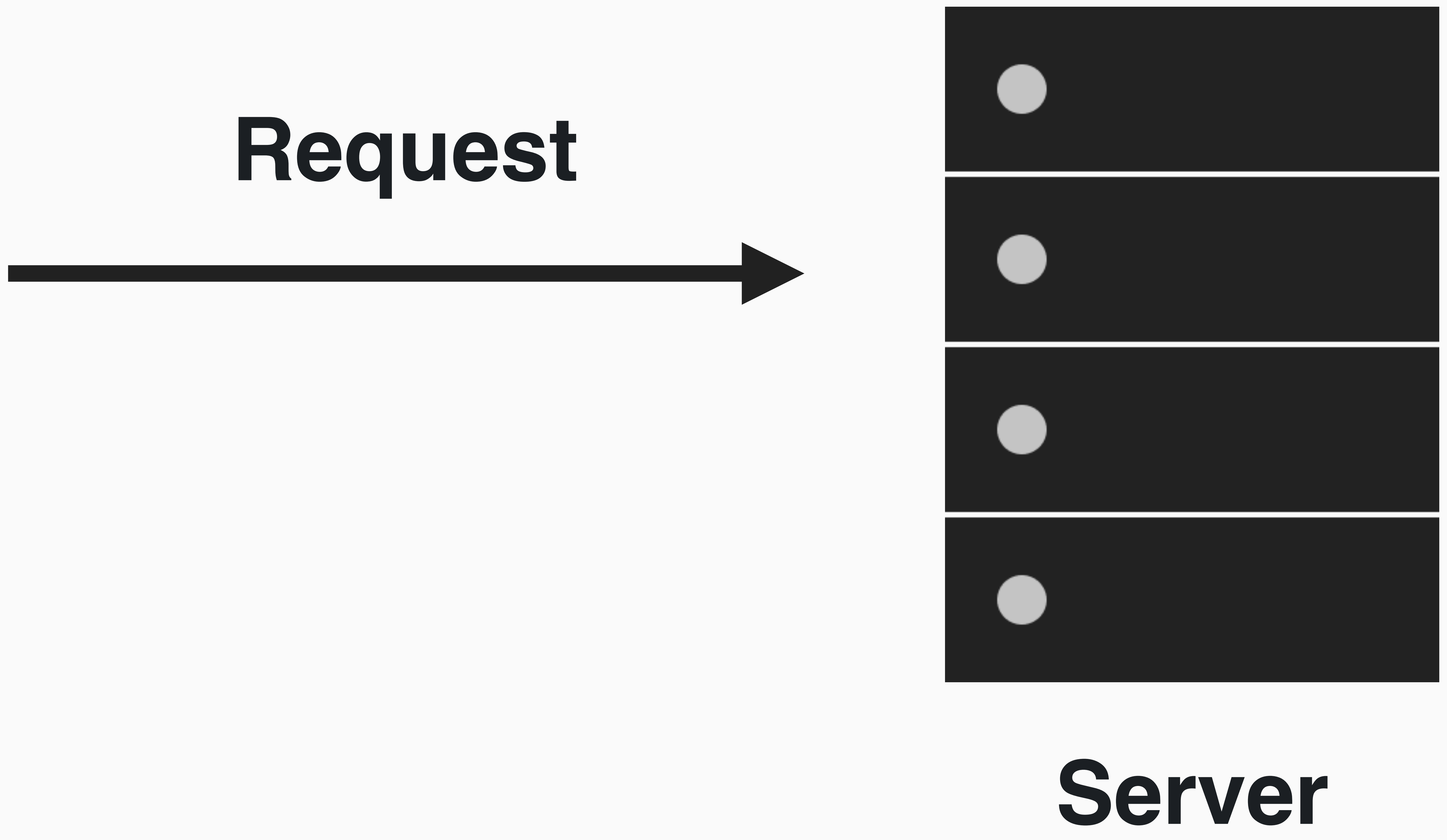
- Connects server to database
- Defines tables and its columns
- Inserts, updates, retrieves, and deletes information from tables

Client
Server


Request
Client
Server



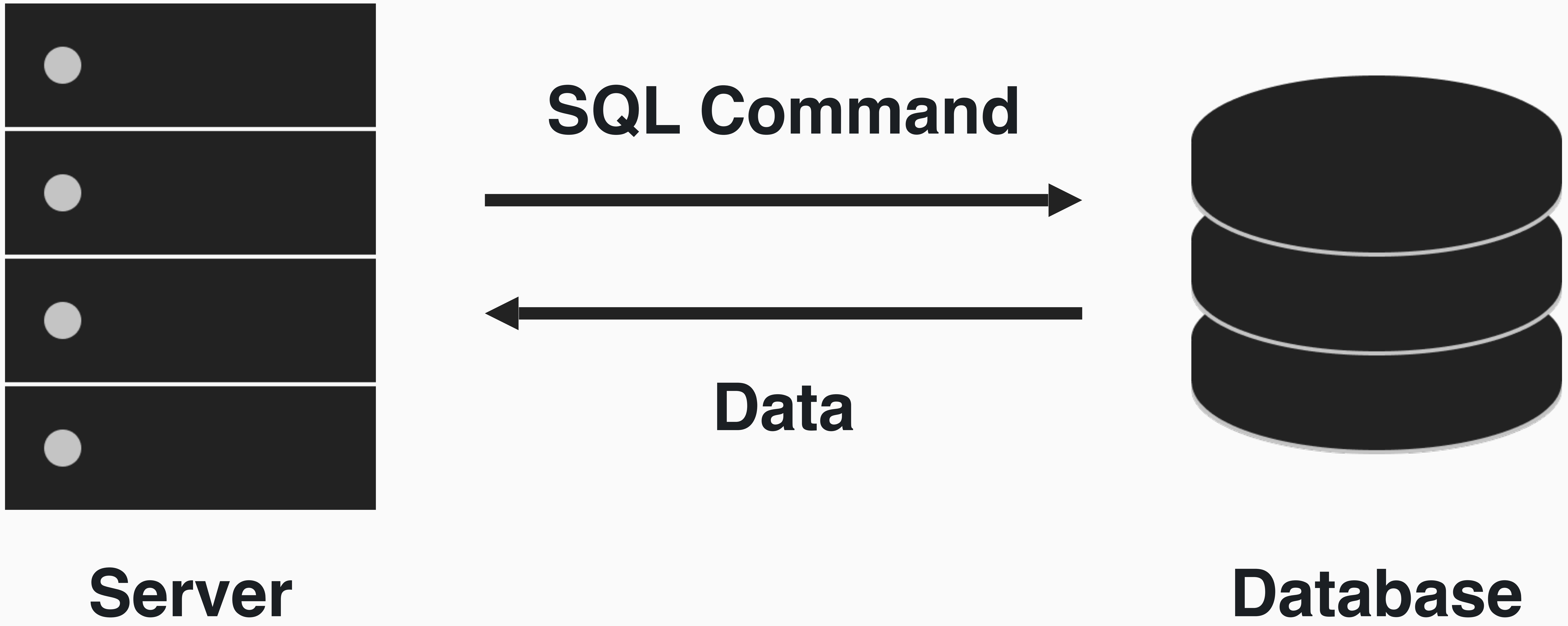
→ Route triggered
Data operation
Return response

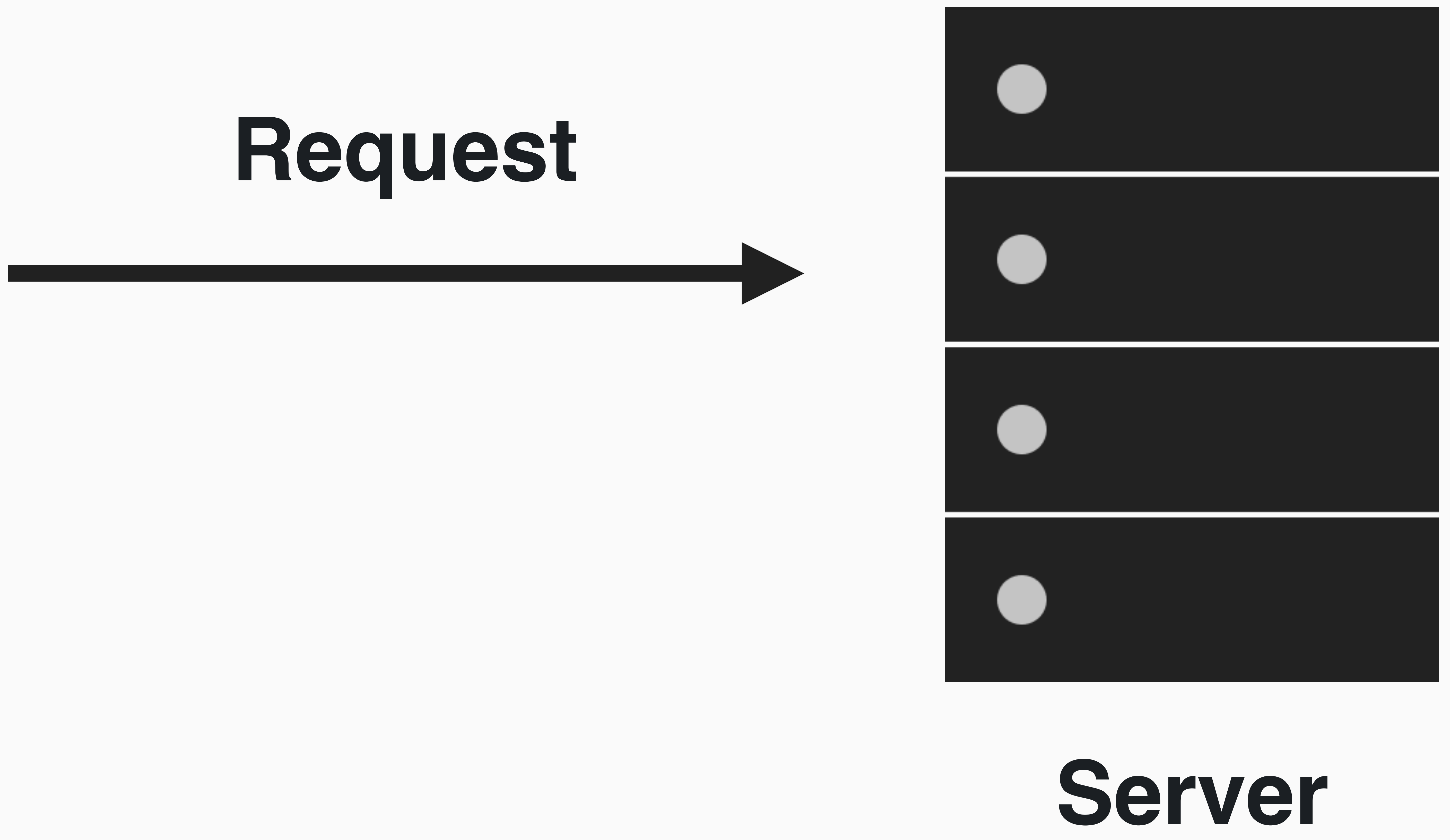


Route triggered

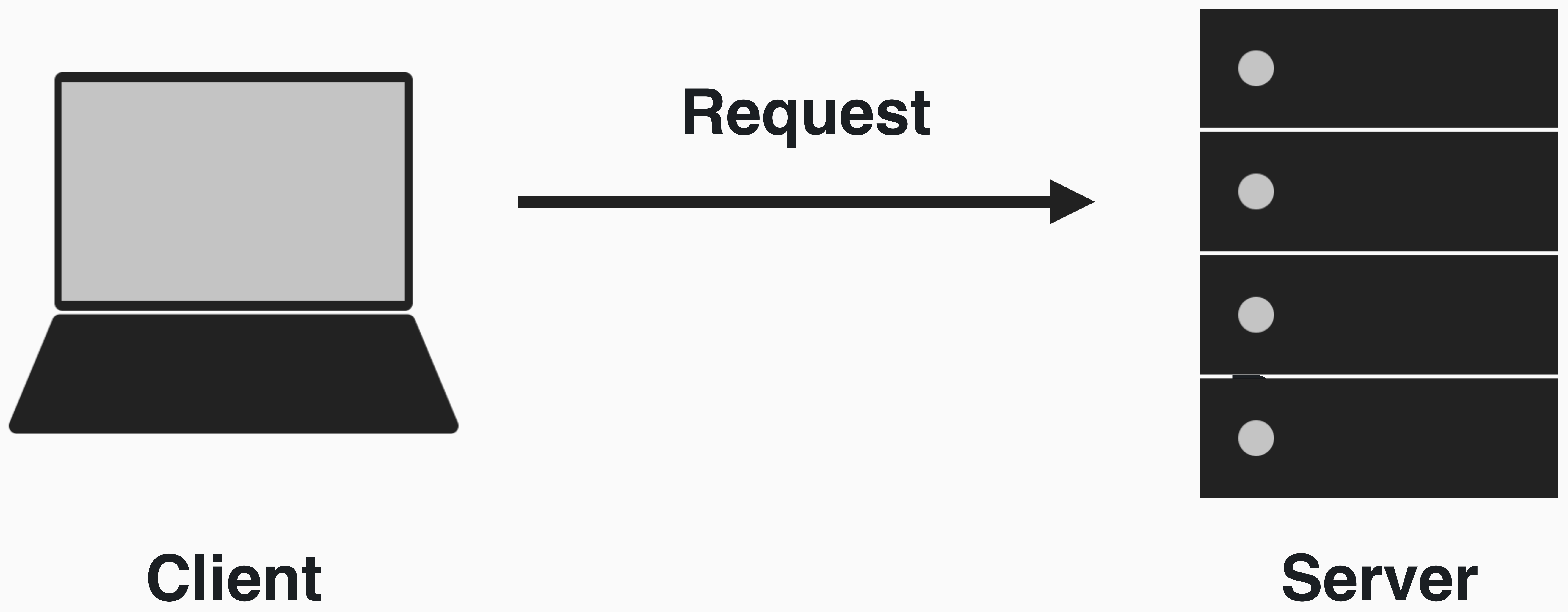
→ Data operation

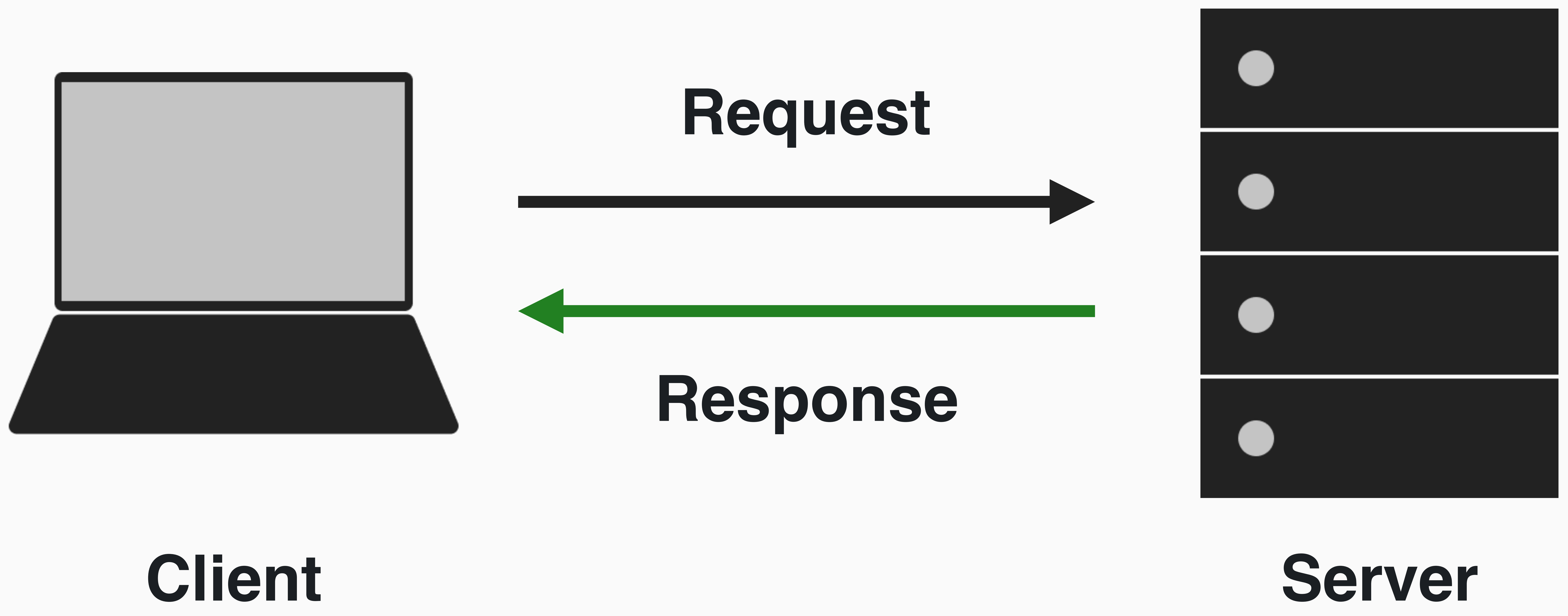
Return response


SQL Command
Data
Server
Database



Route triggered
Data operation
➡ Return response


Request
Client
Server


Request
Response
Client
Server

Demo