

**SQL  
INJECTION**

# SQL --> Structured Query Language

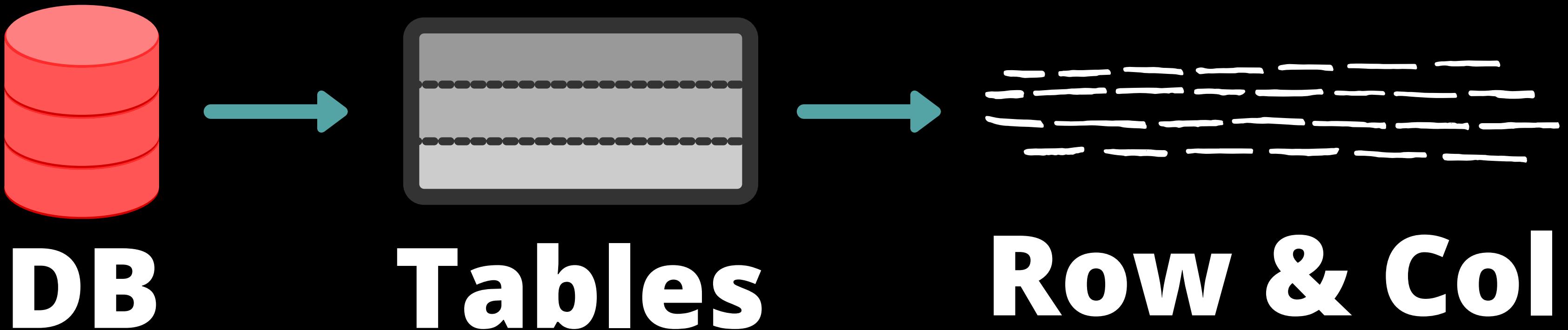


It is a database with tables and values

Provides data from backend to frontend apps



# SQL Struct



Each column specifies an attribute of the DB

There can be many rows based on the app

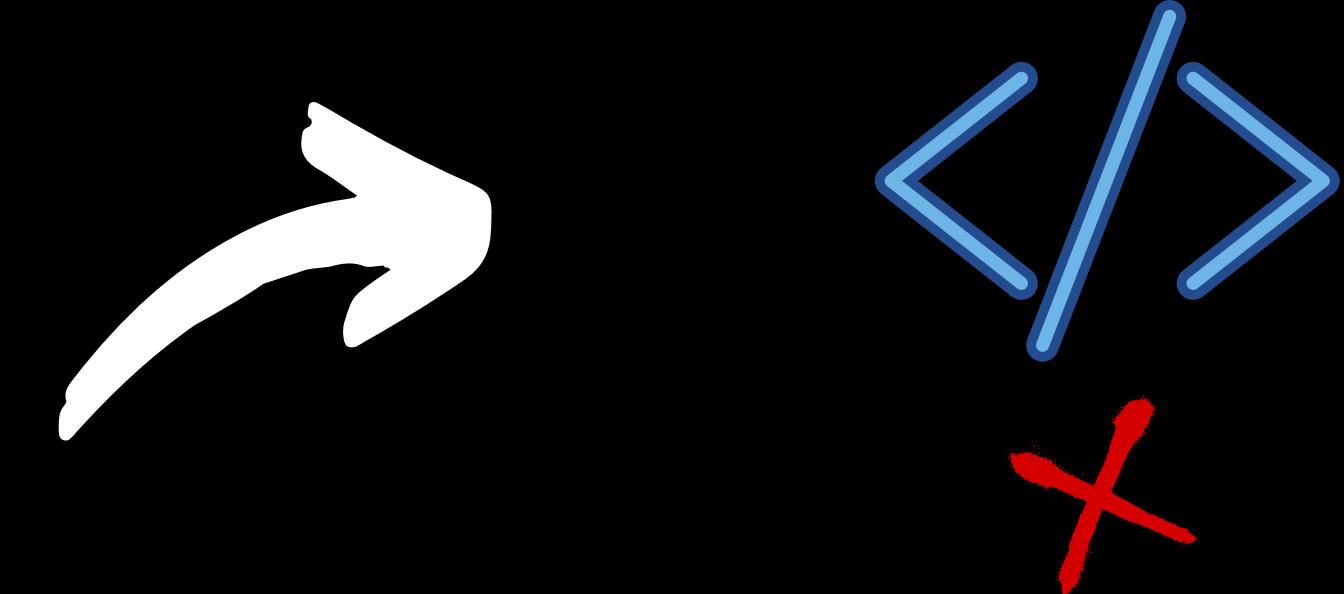
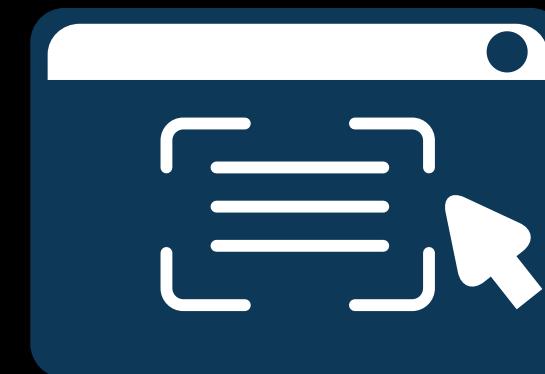
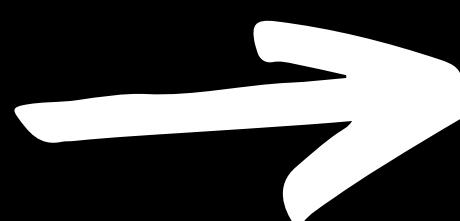
Each row can be referenced by primary key

Tables can be interlinked in SQL using foreign key

Normalization reduces duplicates in the table

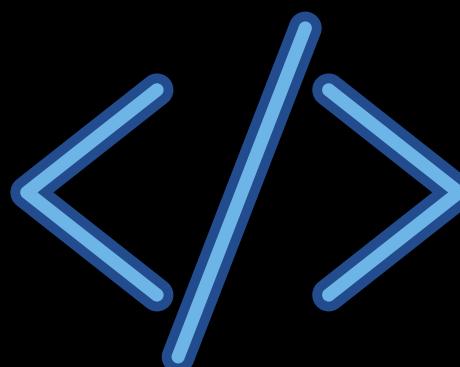
# HOW SQL DB WORKS?

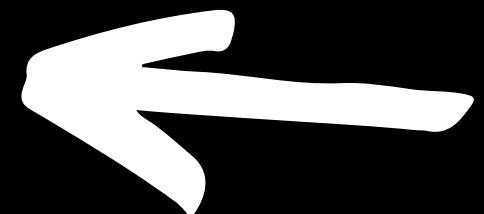
DATA EMBEDDED  
IN SOURCE CODE



performance, size  
security, transparency

DATA CALLED  
FROM DB SERVER

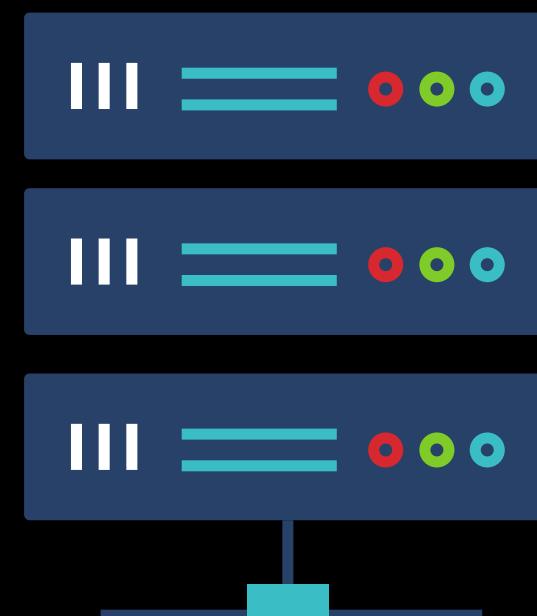




finds row with value = moni  
retrieves all column attributes



username?=moni



finds "users" table  
which contains username

**SQL RUNS ON  
"QUERIES"  
NOT FUNCTIONS  
AND COMMANDS**

# COMMON SQL QUERY

DATA  
DEFINITION

CREATE  
DROP  
ALTER  
TRUNCATE

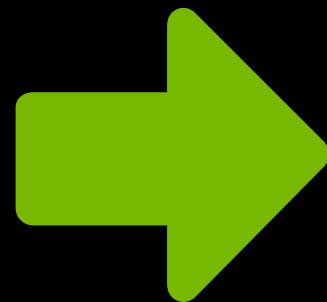
DATA  
MANIPULATION

SELECT  
UPDATE  
INSERT  
DELETE

DATA  
AGGREGATION

SUM  
AVG  
MAX  
MIN  
SO ON..

username?=moni

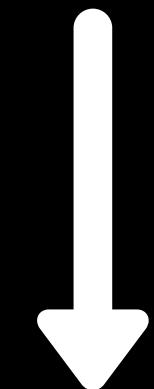


## TABLE

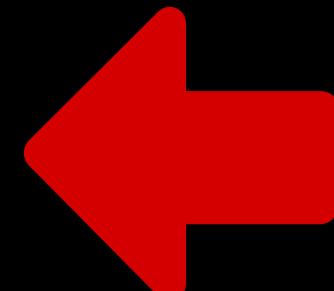
...  
posts  
users  
products

...

selects "users" table

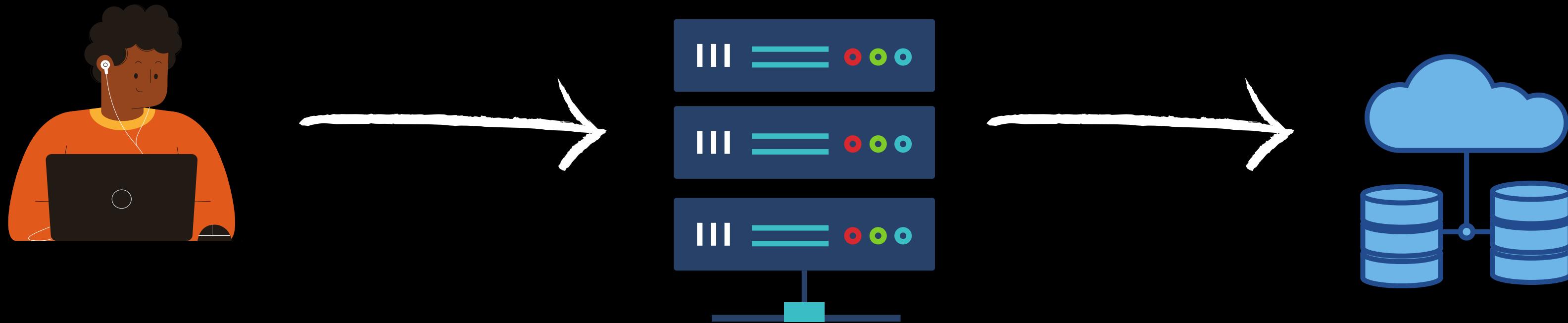


	<b>uid</b>	<b>username</b>	<b>email</b>	<b>password</b>	...
1		abc	a@a.com	abcdefgh	...
2		moni	m@a.com	passwd	...
3		nanba	vn@a.com	nasahack	...



selects row with  
"username=moni"

**SQL**  
**WORKING**



# VALUE + QUERY == RESULT

## DATA

user input field  
form data  
url parameters  
etc..

## FUNCTION

retrieve  
update  
post  
delete & ..

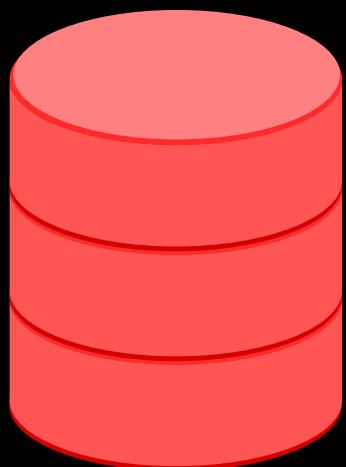


## SQL QUERY

works in DB  
based on  
"values" from "DATA"  
"queries" from "FUNCTION"



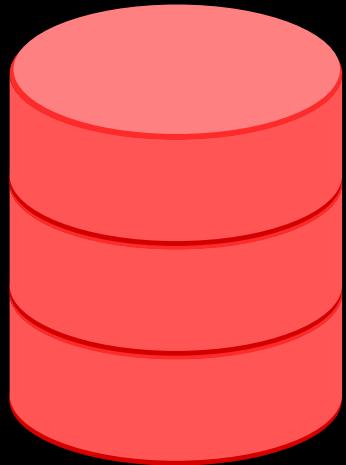
correct  
input data



SQL Query  
produces  
**"Expected  
Correct  
Result To User"**

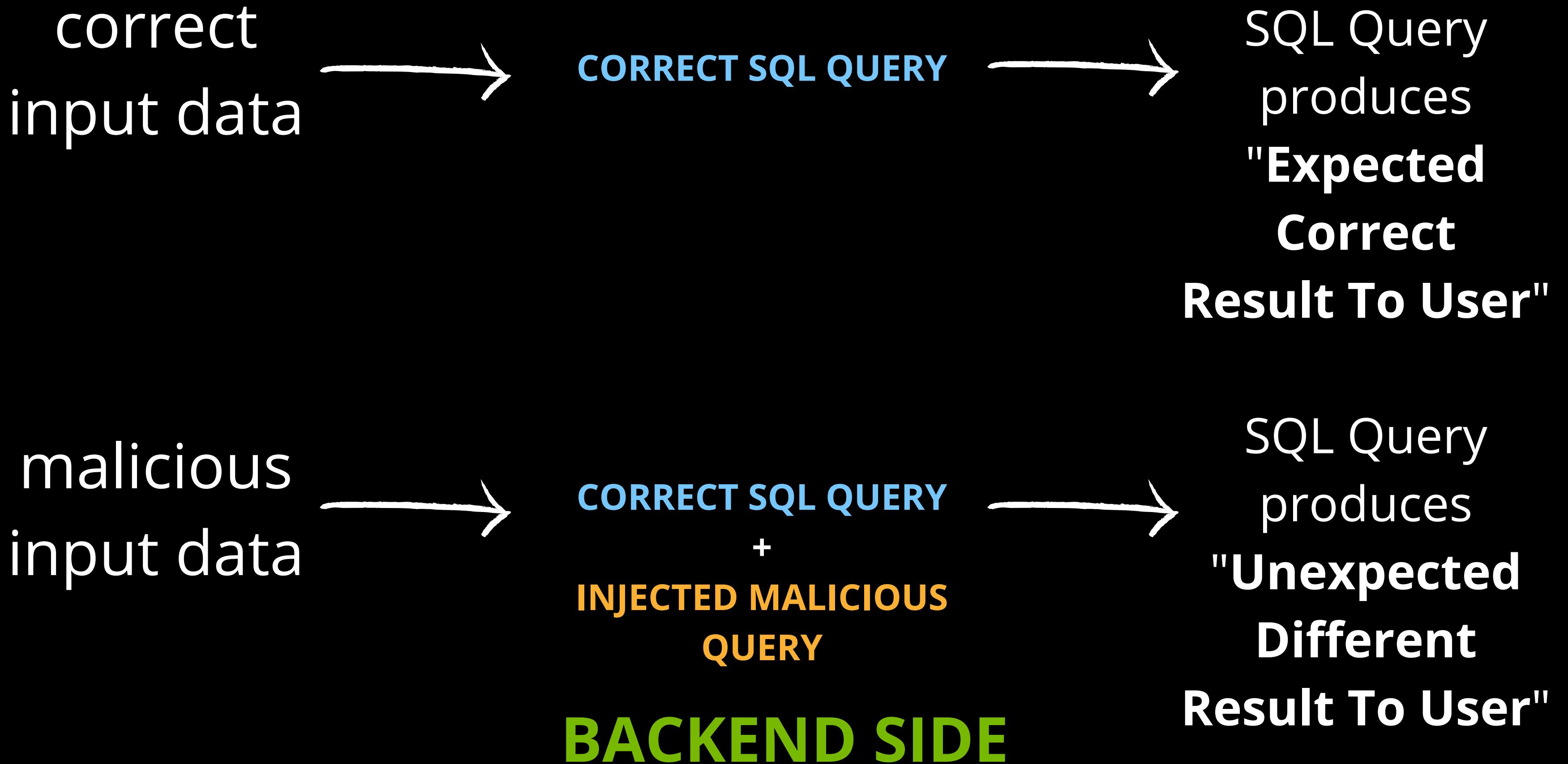


malicious  
input data



SQL Query  
produces  
**"Unexpected  
Different  
Result To User"**

**FRONTEND SIDE**



This "**UNEXPECTED RESULT**" caused due to the injection of malicious input via SQL query into the database table is known as "**SQL Injection**"



# PREVENTION?

- Validate user inputs
- Sanitize data with limited special characters
- Prefer whitelist over blacklist
- Limit privileges and read-access
- Do not link to other valuable tables
- Scanning and updating
- Usage of WAF

# THANK YOU

MYSELF



@aidenpearce369