SQL🔥

SQL🔥

The language we use to talk with databases

SQL🔥

The language we use to talk with databases

Structured Query Language

SQL🔥

The language we use to talk with databases

Structured Query Language

Born in the 1970s at IBM.

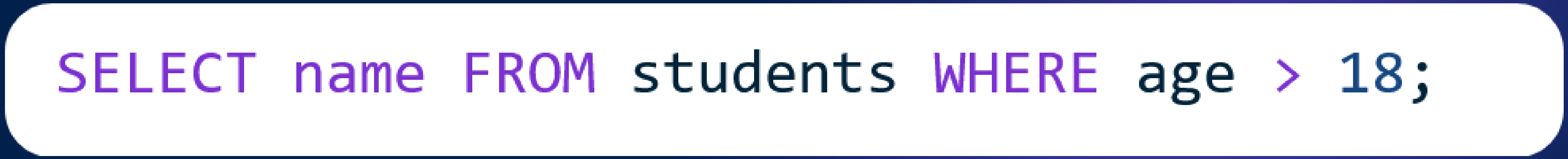
Became a standard in 1986

Supports Every Modern Databases

Declarative

Declarative

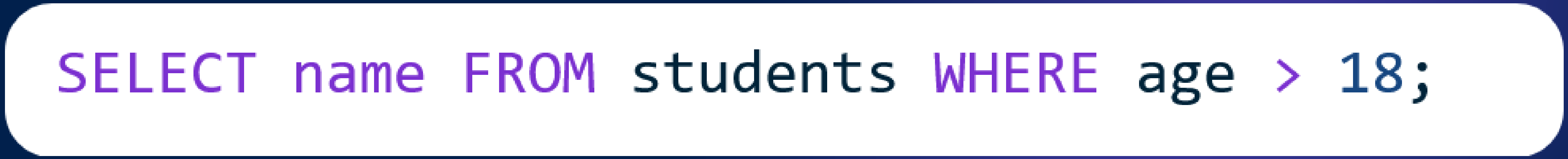
You tell the database what you want, not how to do it.



```sql
SELECT name FROM students WHERE age > 18;
```

Declarative

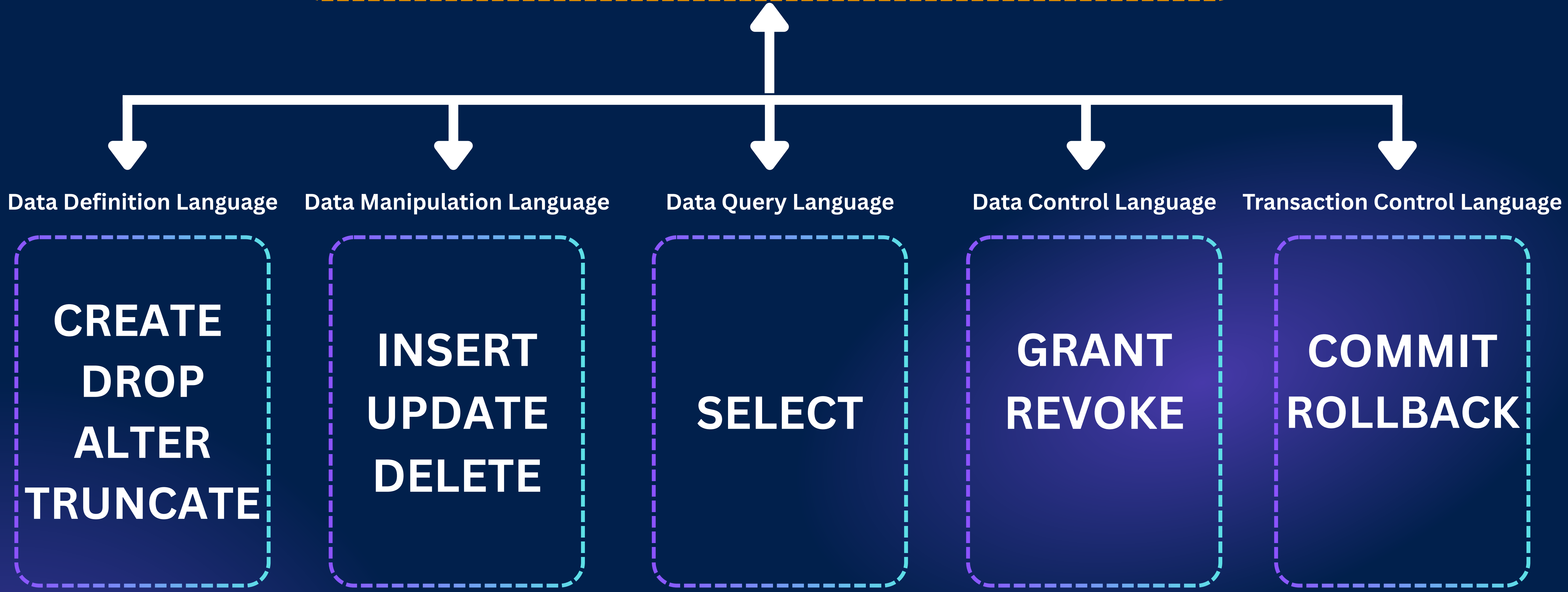
You tell the database what you want, not how to do it.



```sql
SELECT name FROM students WHERE age > 18;
```

SQL Statement

Categories of SQL Commands



Data Definition Language

CREATE
DROP
ALTER
TRUNCATE

Data Manipulation Language

INSERT
UPDATE
DELETE

Data Query Language

SELECT

Data Control Language

GRANT
REVOKE

Transaction Control Language

COMMIT
ROLLBACK

So, SQL is a declarative language to interact with databases. It’s old but gold, and still the backbone of all modern data systems.

It’s also powering the future with AI🔥.

Normal Forms

2NF

Instructor

c_id	Instructor
1	Prof. Smith
2	Prof. Johnson
3	Prof. Adams

Course

c_id	c_name
1	Math
2	Science
3	History

Lossy Decomposition

Data Types

id (SERIAL)	employee_id (INTEGER)	name (VARCHAR(50))	dob (DATE)	is_active (BOOLEAN)
1	4560	John	1990-05-15	true
2	8962	Doe	1985-08-22	false

Data Types

id (SERIAL)	employee_id (INTEGER)	name (VARCHAR(50))	dob (DATE)	is_active (BOOLEAN)
1	4560	John	1990-05-15	true
2	8962	Doe	1985-08-22	false

Data Accuracy

Memory Efficiency

Performance ⚡

Clarity & Constraints

Data Types

Boolean	Numbers	Binary
Date/Time	Json	Character
UUID	Array	XML

Data Types

Boolean	Numbers	Binary
Date/Time	Json	Character
UUID	Array	XML

Boolean

true , false, null

Integer

Data Type	Storage	Range	Use Case
SMALLINT (int2)	2 bytes	-32,768 to +32,767	Small numbers (like age, quantity)
INTEGER (int4)	4 bytes	~ -2B to +2B	Default choice for whole numbers
BIGINT (int8)	8 bytes	~ -9 quintillion to +9 quintillion	Very large numbers (IDs, counters)
REAL (float4)	4 bytes	~6 decimal digits precision	Approximate values (e.g., sensor data)
DOUBLE PRECISION (float8)	8 bytes	~15 decimal digits precision	Higher precision calculations
NUMERIC / DECIMAL	Variable	User-defined precision (exact)	Money, financial calculations
SERIAL	4 bytes (auto-increment integer)	1 to 2,147,483,647	Auto-incrementing IDs, primary keys

Character

Data Type	Storage	Length	Use Case
CHAR(n)	n bytes	Fixed length n	When you know the exact length (like country codes: 'USA')
VARCHAR(n)	Variable	Up to n characters	Flexible length but with a max limit (like usernames, emails)
TEXT	Variable	Unlimited	Long text, descriptions, comments

Date

Data Type	Example
DATE	'1980-12-20'
TIME	'14:30:00'
TIMETZ	'14:30:00+06'
TIMESTAMP	'2025-08-29 14:30:00'
TIMESTAMPTZ	'2025-08-29 14:30:00+06'
INTERVAL	'3 days 4 hours'

20-Dec-1980

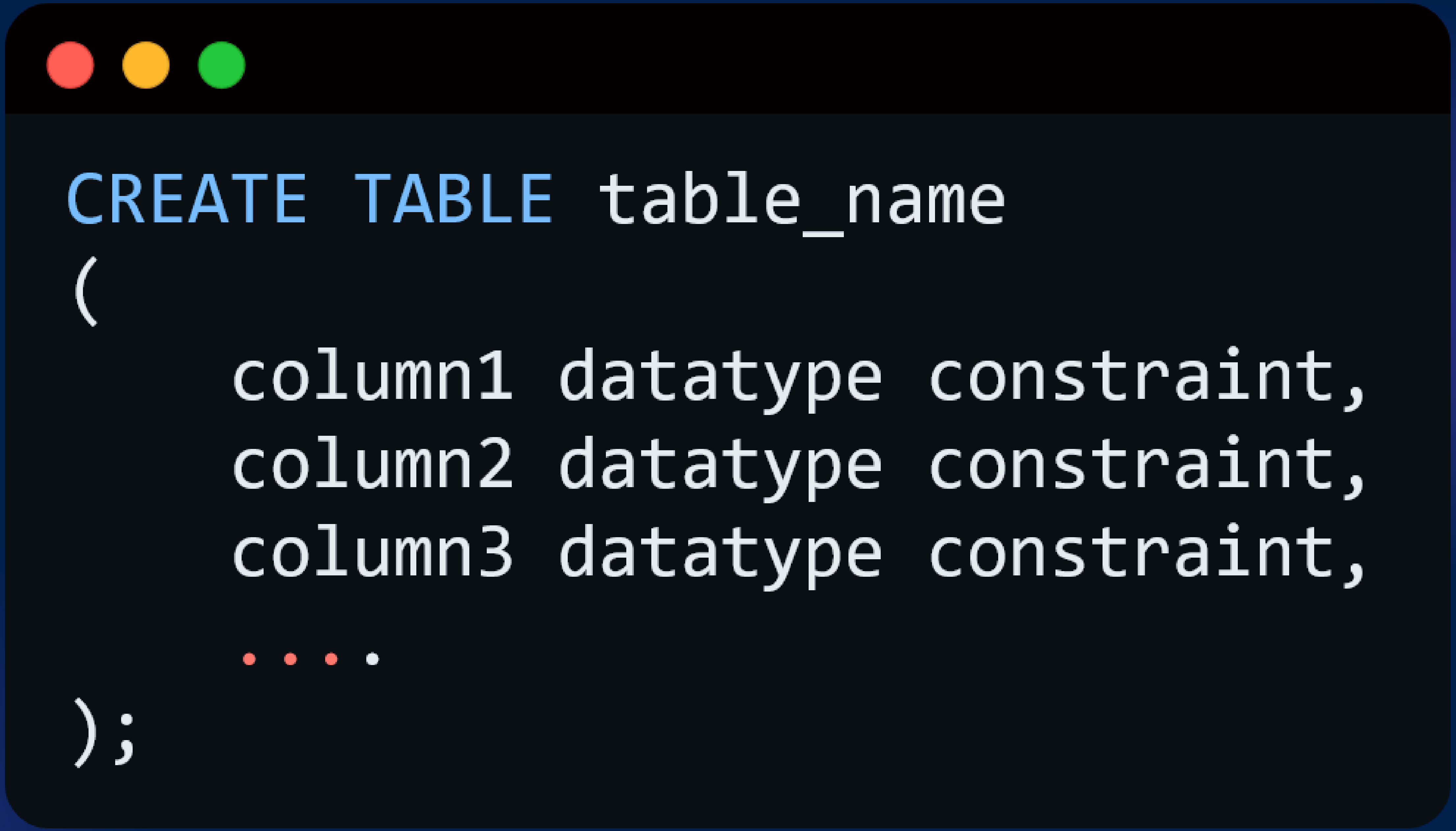
Dec-20-1980

UUID

Data Type	Example
UUID	'550e8400-e29b-41d4-a716-446655440000'

UUID stands for Universally Unique Identifier.

Creating Table

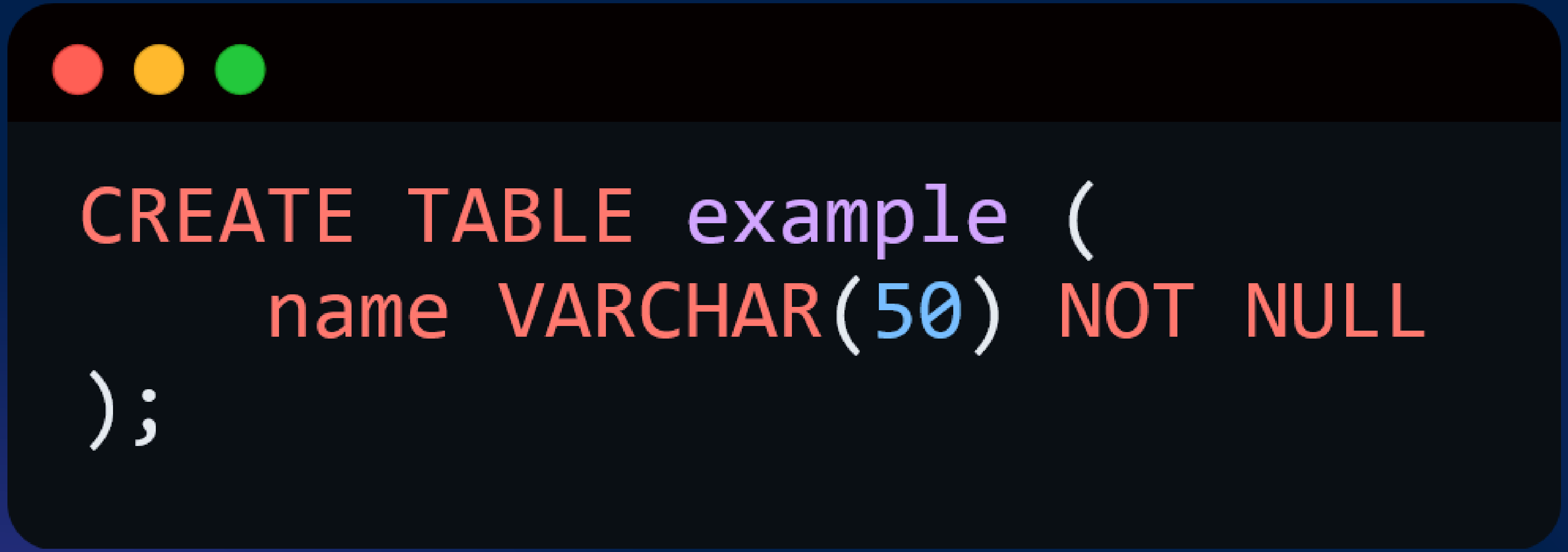


```
CREATE TABLE table_name
(
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

Column Constraints

Column Constraints

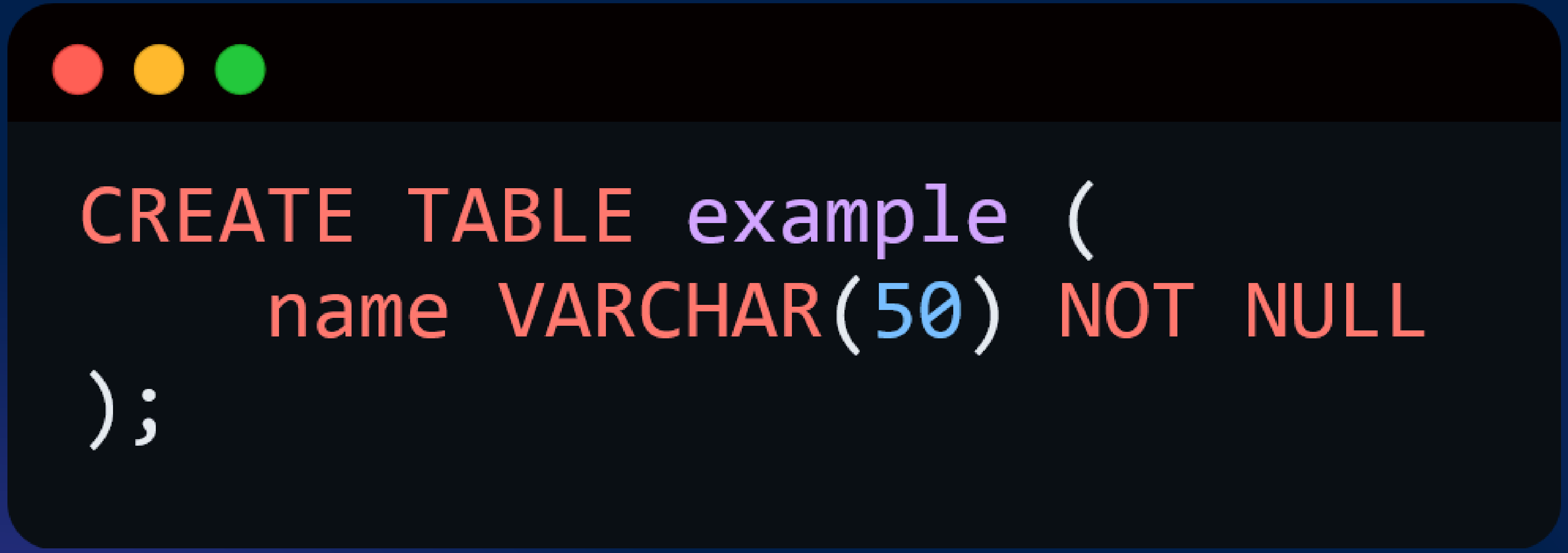
NOT NULL



```
CREATE TABLE example (
    name VARCHAR(50) NOT NULL
);
```

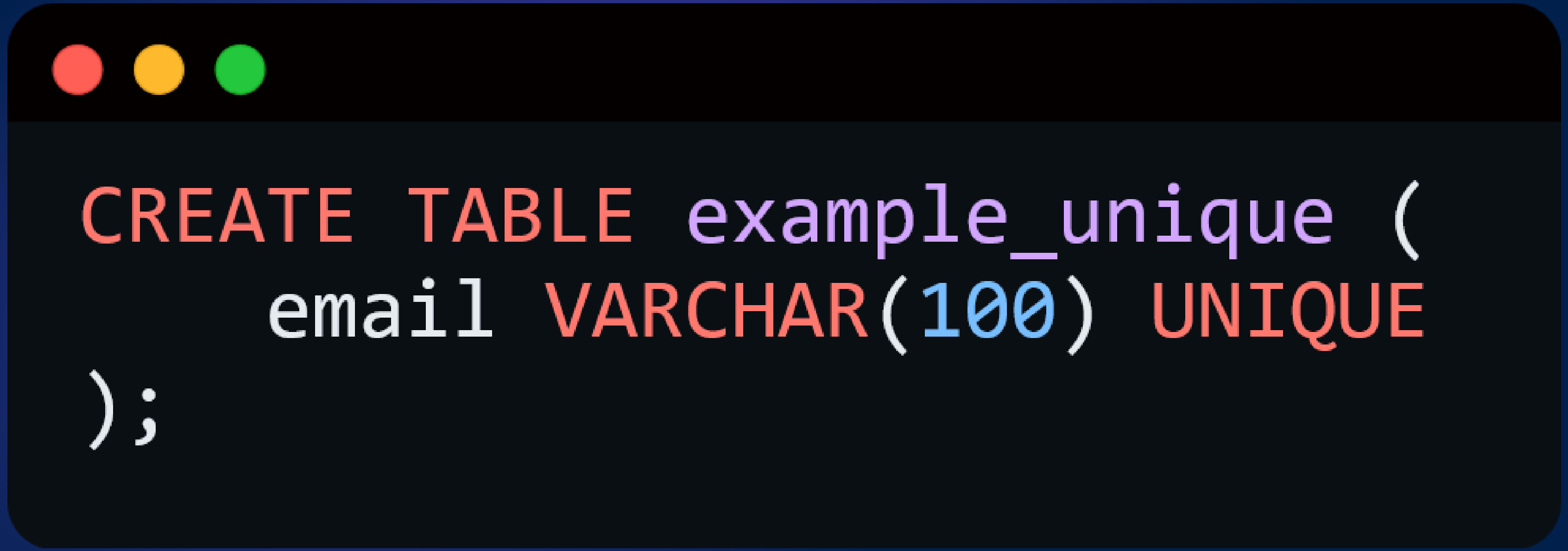
Column Constraints

NOT NULL



```
CREATE TABLE example (
    name VARCHAR(50) NOT NULL
);
```

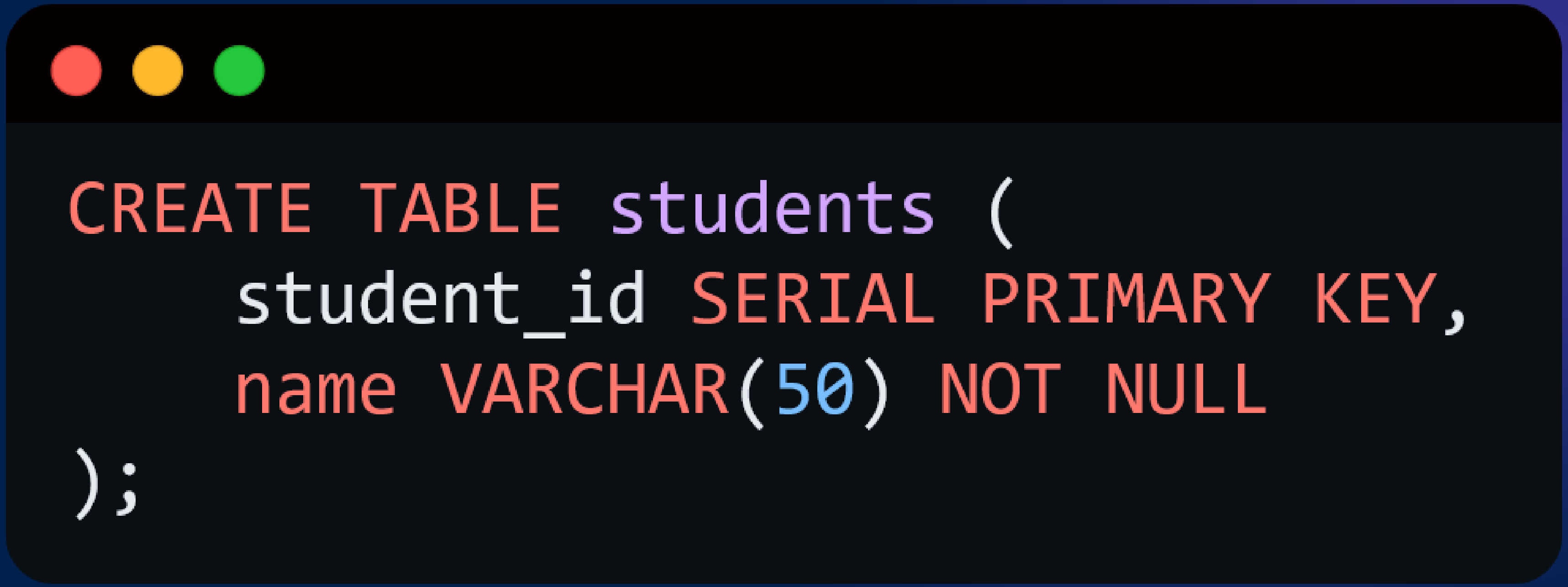
UNIQUE



```
CREATE TABLE example_unique (
    email VARCHAR(100) UNIQUE
);
```

Column Constraints

Primary Key

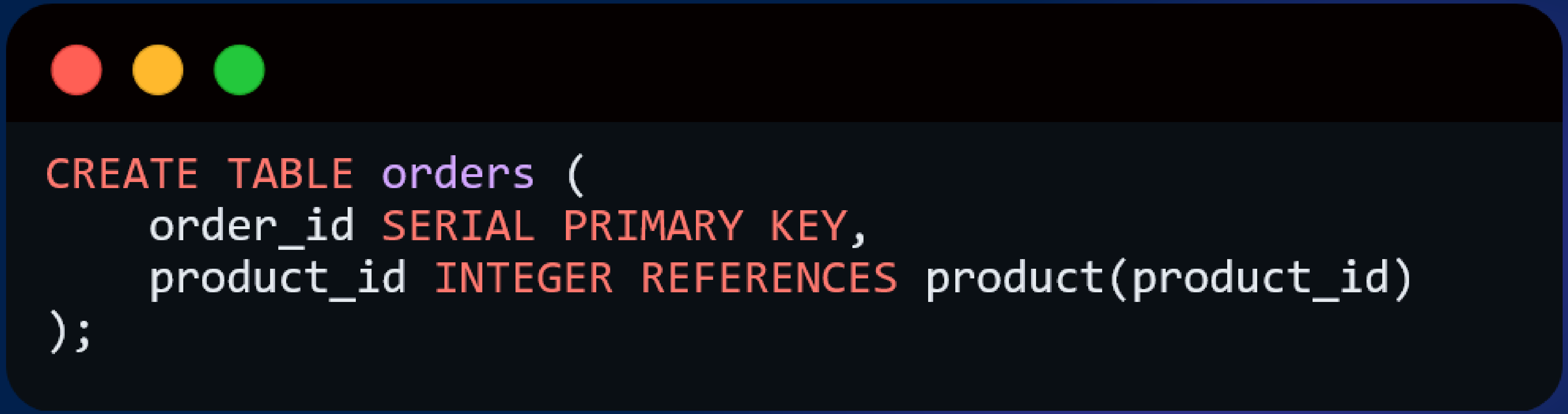


```
CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL
);
```

Primary Key = must be unique + cannot be null

Column Constraints

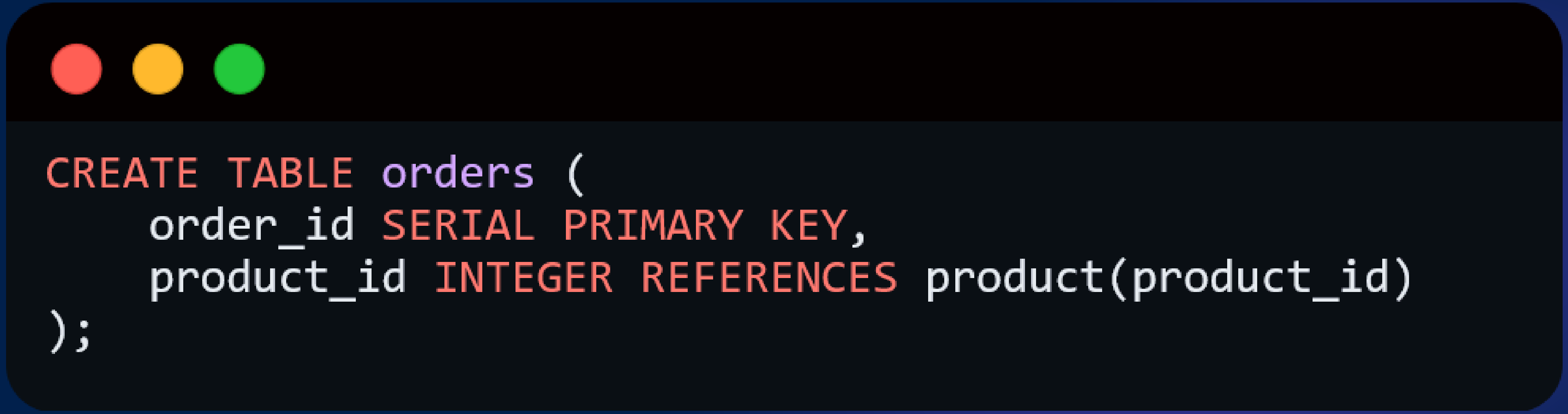
Foreign Key



```
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    product_id INTEGER REFERENCES product(product_id)
);
```

Column Constraints

Foreign Key



```
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    product_id INTEGER REFERENCES product(product_id)
);
```

Product

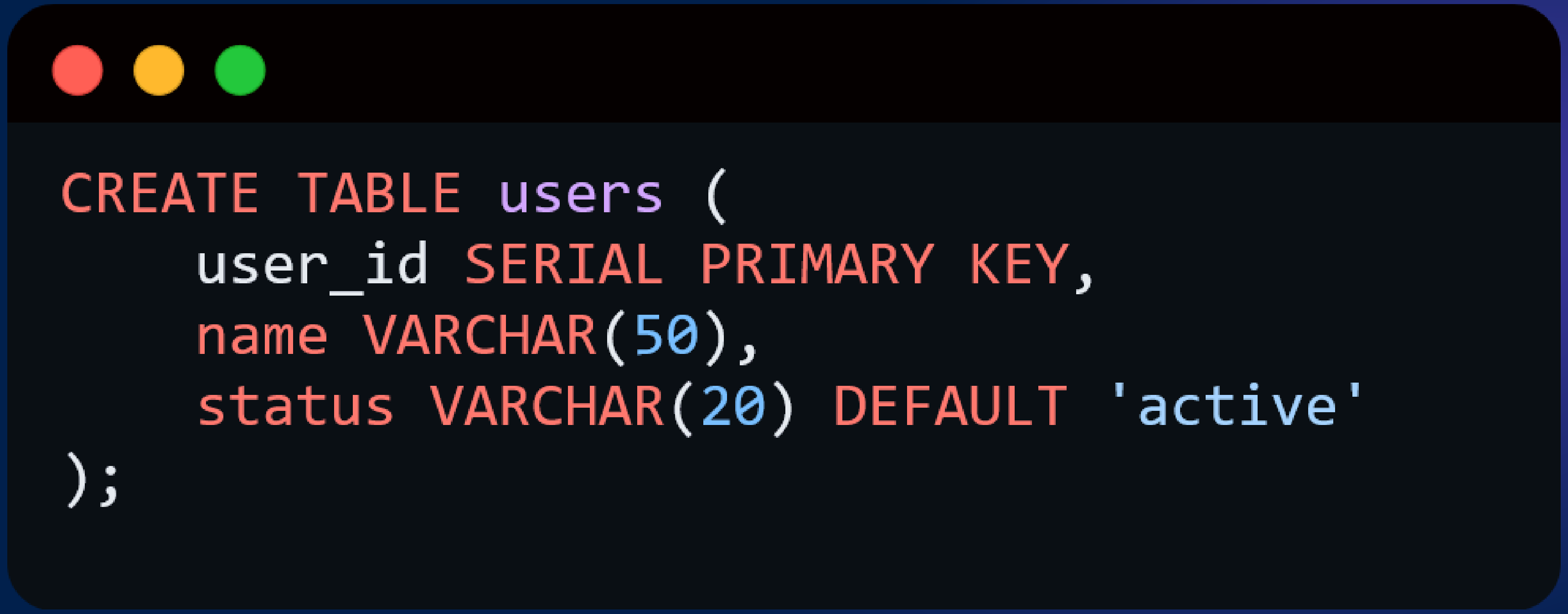
product_id	product_title
1	shoe
2	t-shirt

Order

order_id	prod_id
1	1
2	2

Column Constraints

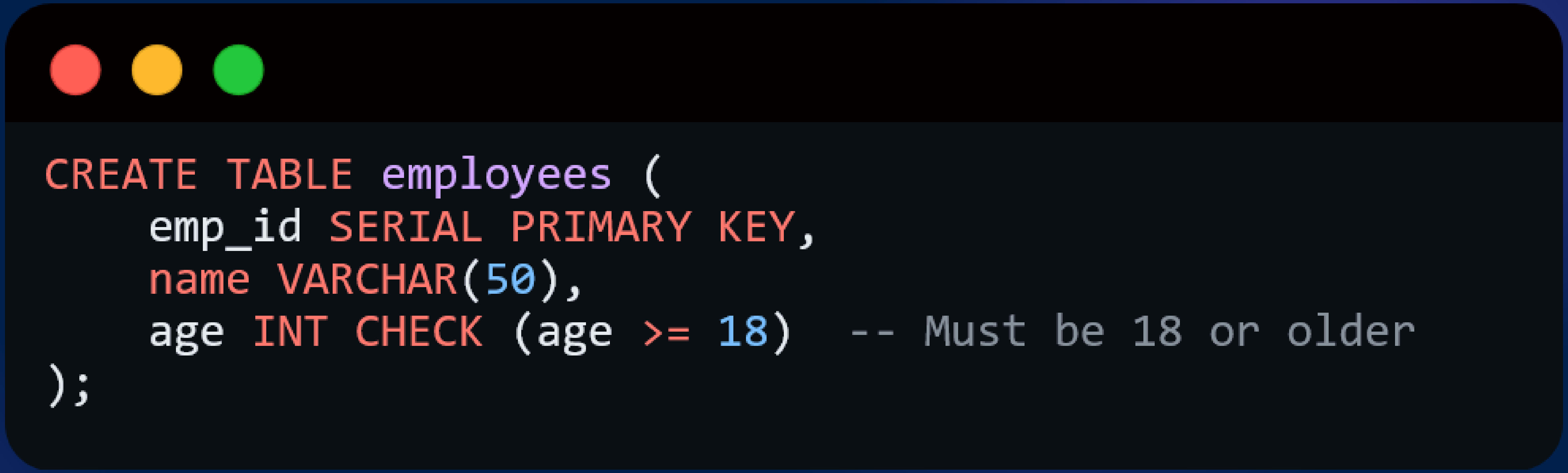
DEFAULT



```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    status VARCHAR(20) DEFAULT 'active'
);
```

Column Constraints

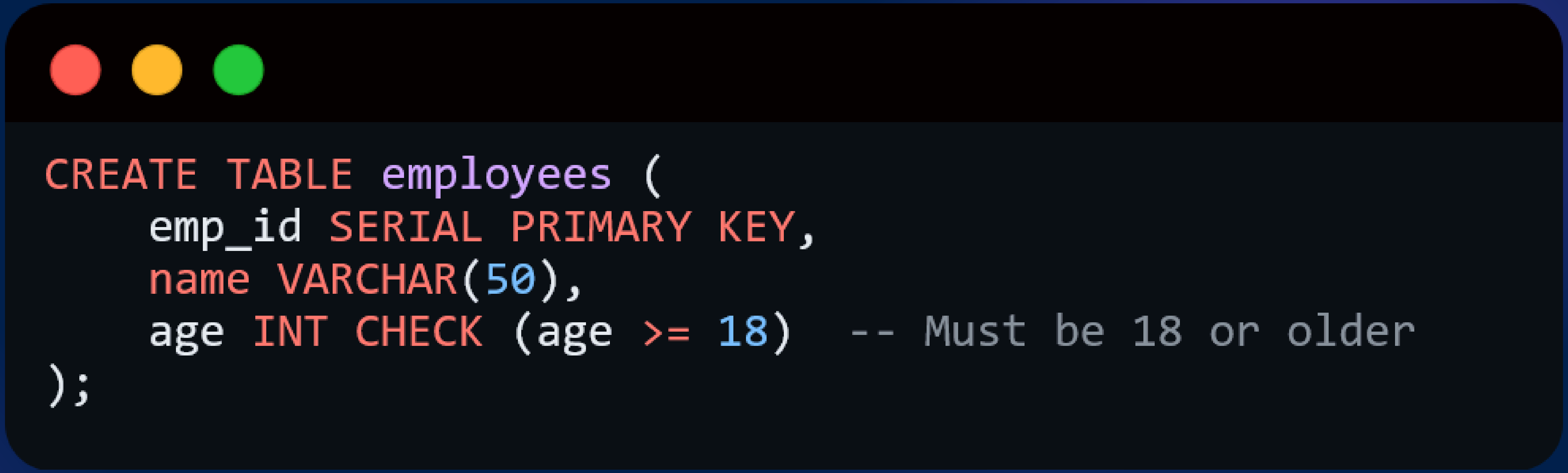
CHECK



```
CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT CHECK (age >= 18)  -- Must be 18 or older
);
```

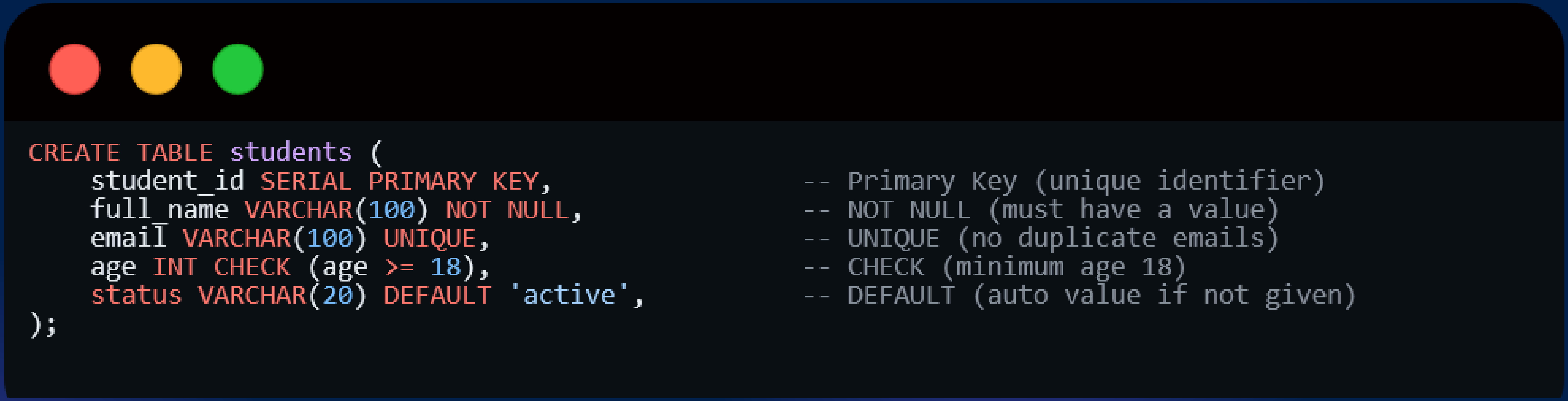
Column Constraints

CHECK



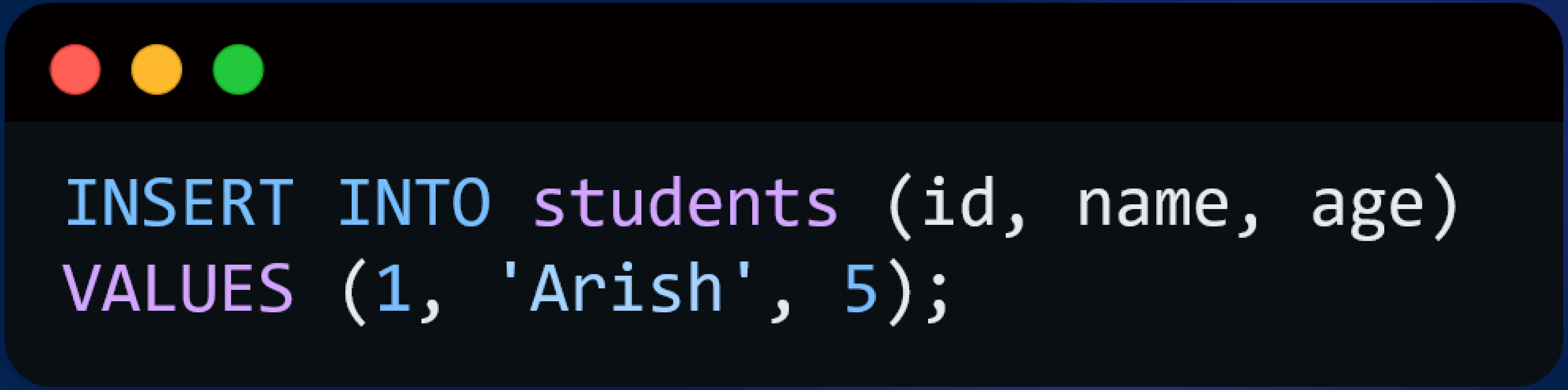
```
CREATE TABLE employees (
    emp_id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT CHECK (age >= 18)  -- Must be 18 or older
);
```

Column Constraints



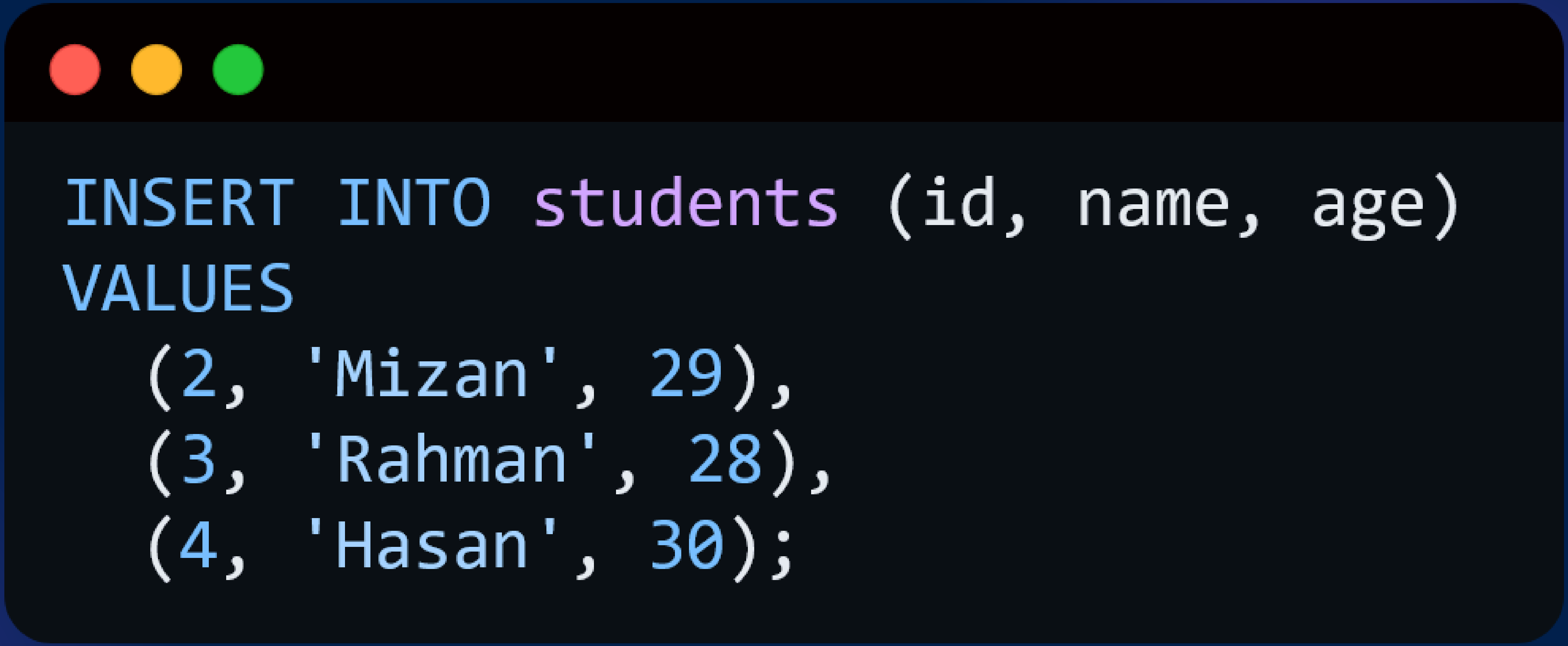
```sql
CREATE TABLE students (
    student_id SERIAL PRIMARY KEY,                -- Primary Key (unique identifier)
    full_name VARCHAR(100) NOT NULL,              -- NOT NULL (must have a value)
    email VARCHAR(100) UNIQUE,                    -- UNIQUE (no duplicate emails)
    age INT CHECK (age >= 18),                    -- CHECK (minimum age 18)
    status VARCHAR(20) DEFAULT 'active',          -- DEFAULT (auto value if not given)
);
```

Single-Row Insert



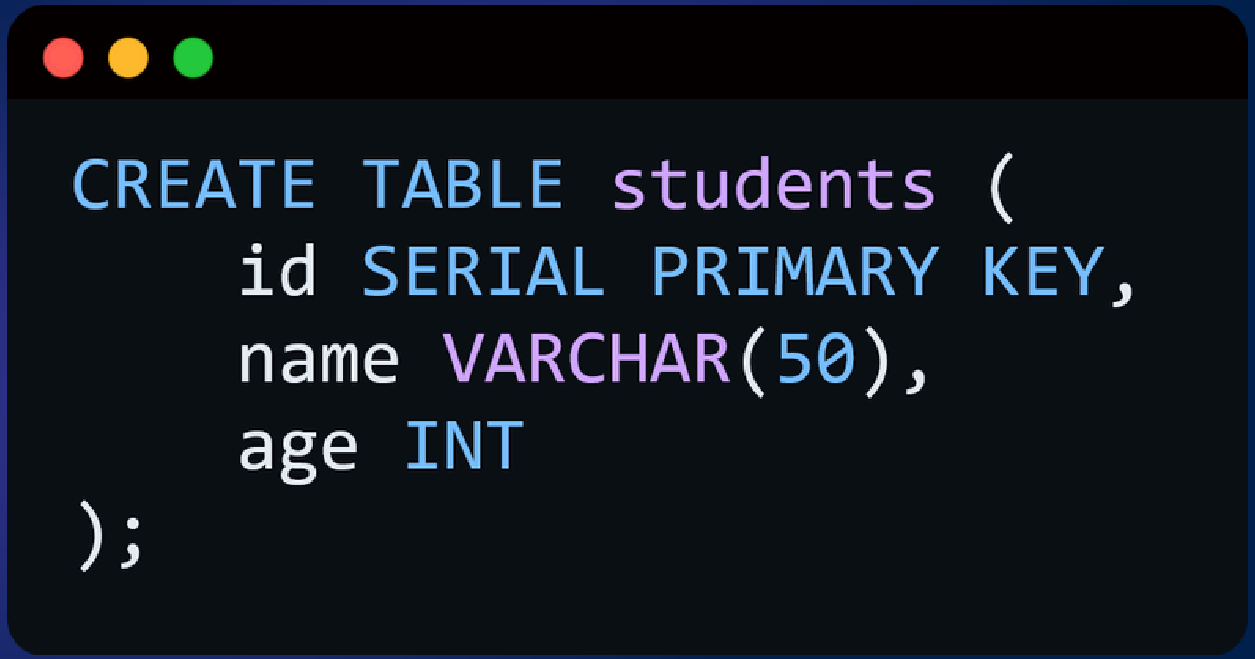
```sql
INSERT INTO students (id, name, age)
VALUES (1, 'Arish', 5);
```

Multi-Row Insert



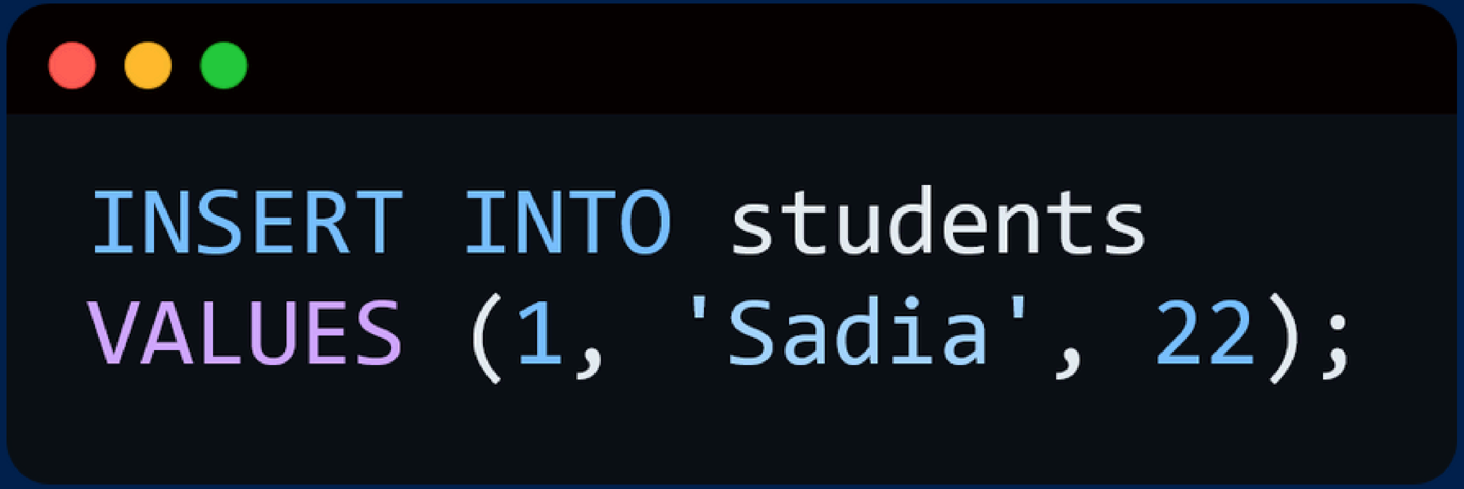
```
INSERT INTO students (id, name, age)
VALUES
  (2, 'Mizan', 29),
  (3, 'Rahman', 28),
  (4, 'Hasan', 30);
```

Without Column List

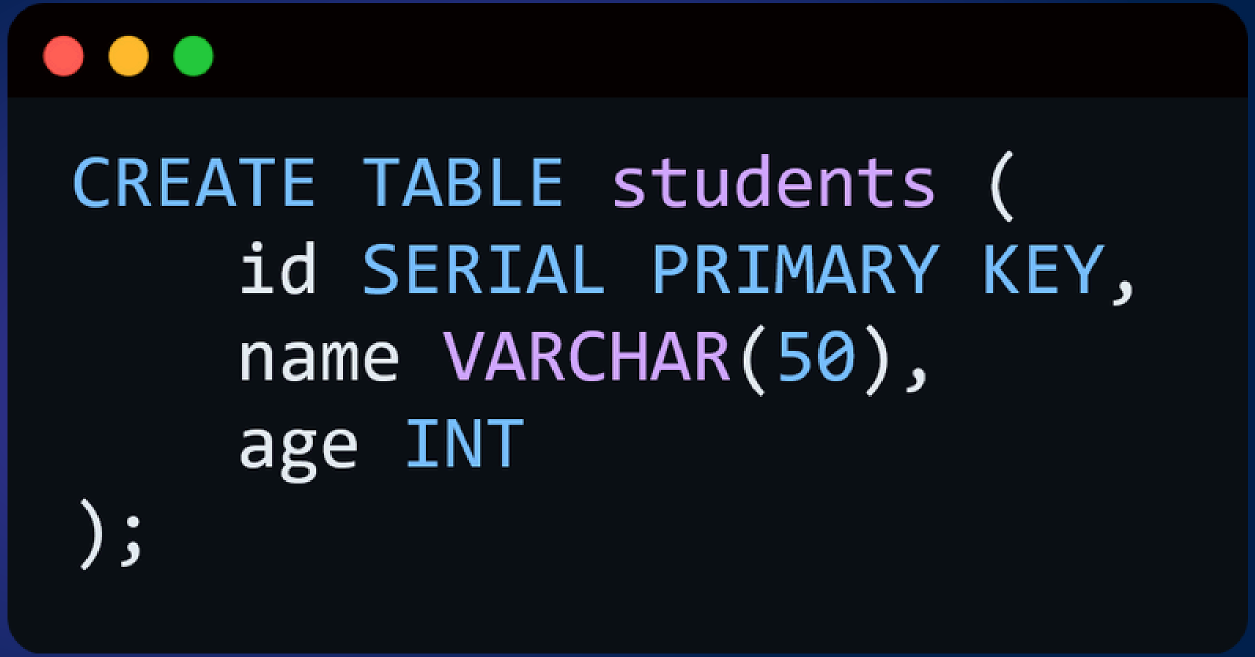


```
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

Without Column List

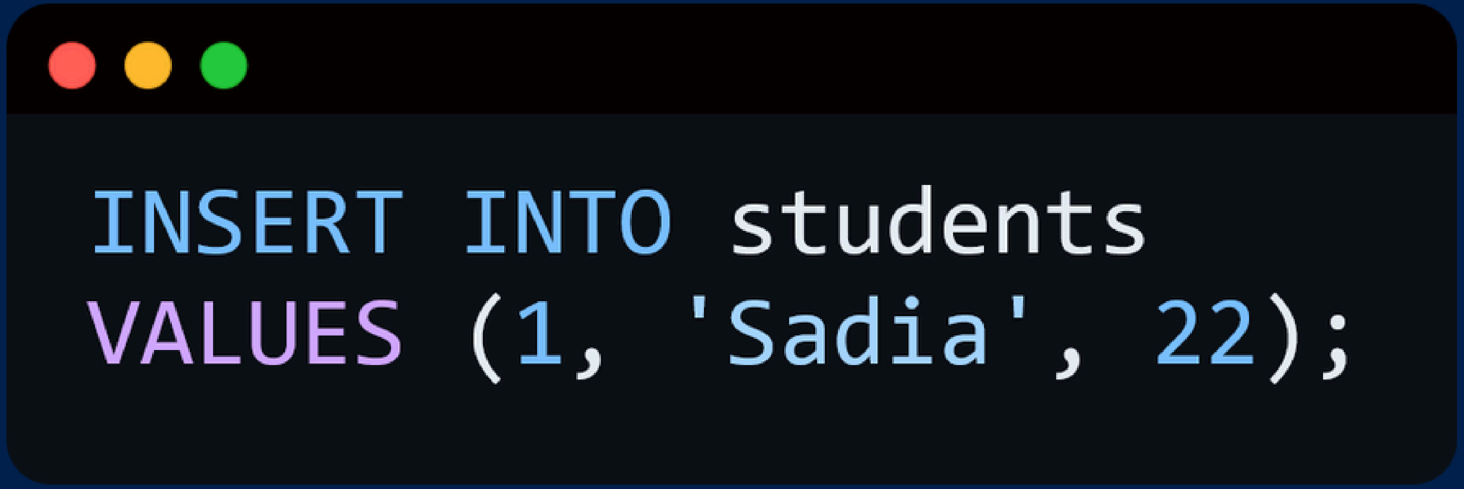


```sql
INSERT INTO students
VALUES (1, 'Sadia', 22);
```

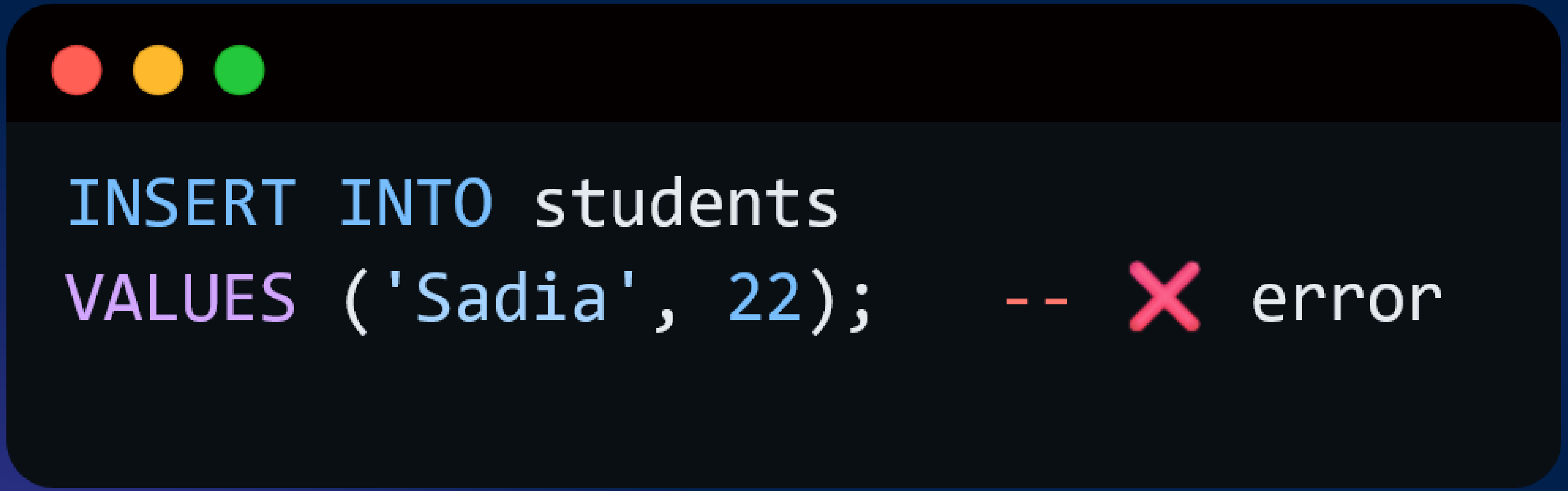


```sql
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

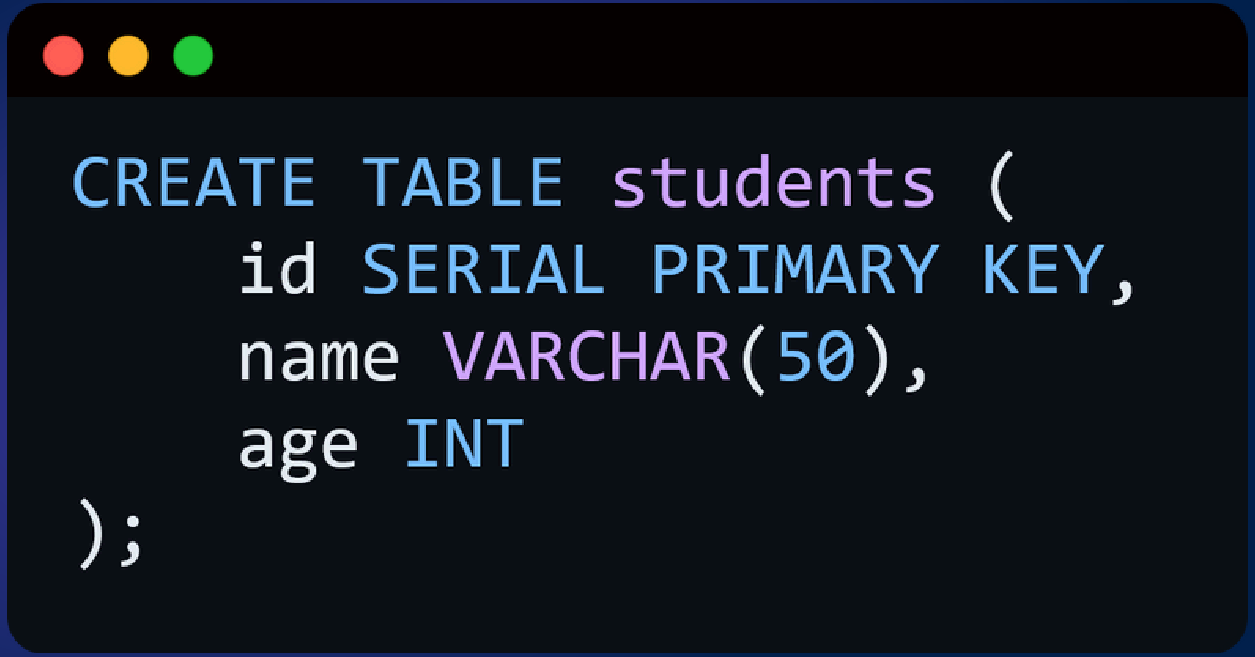
Without Column List



```sql
INSERT INTO students
VALUES (1, 'Sadia', 22);
```

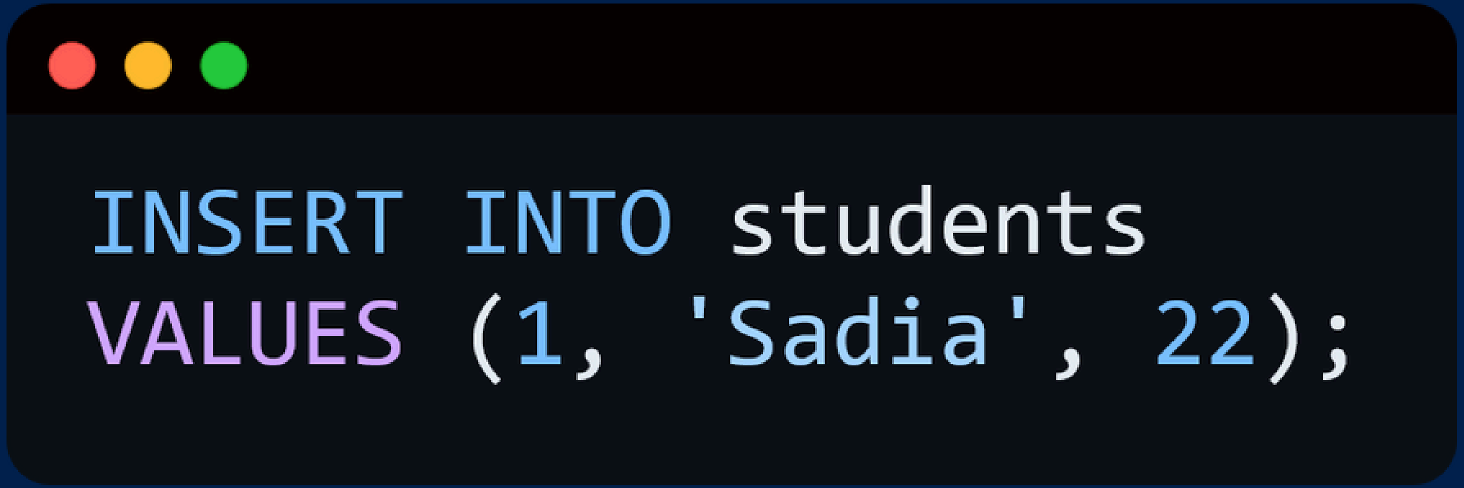


```sql
INSERT INTO students
VALUES ('Sadia', 22);    -- ❌ error
```

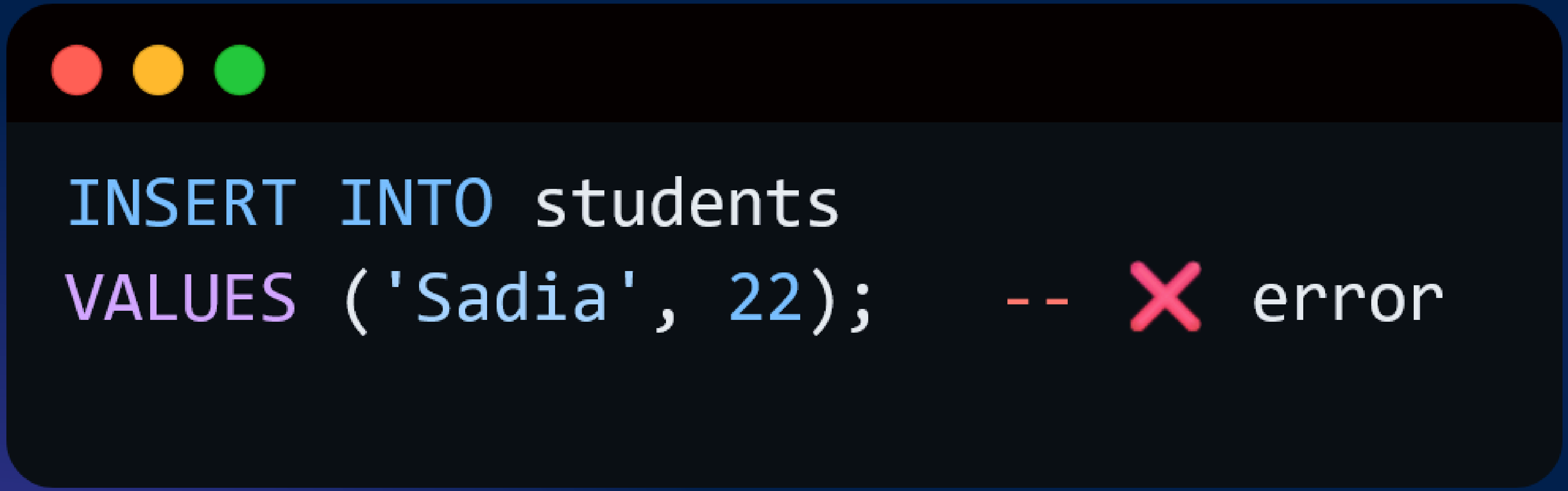


```sql
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```

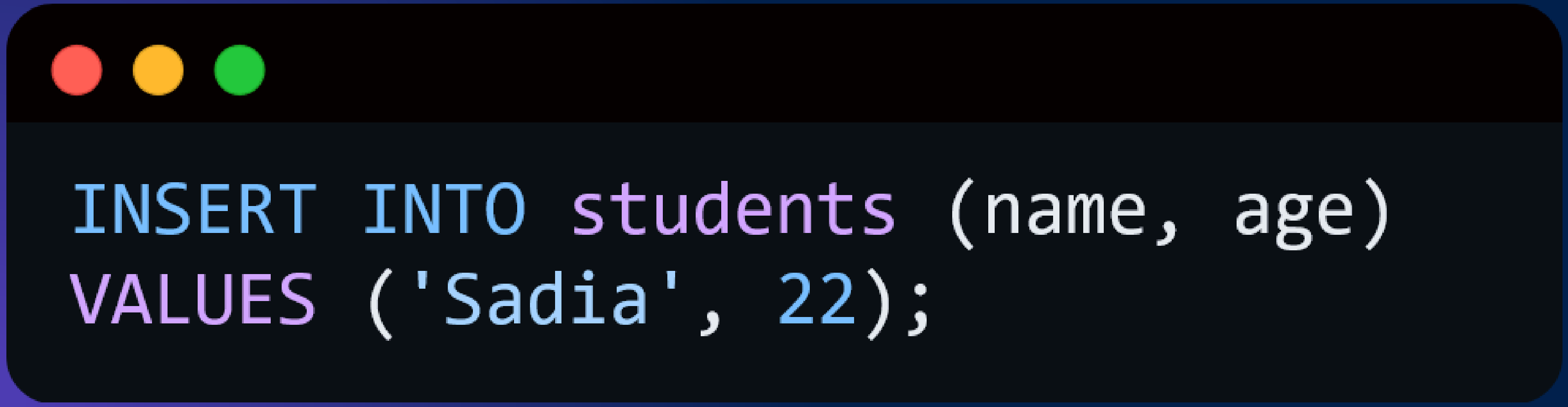
Without Column List



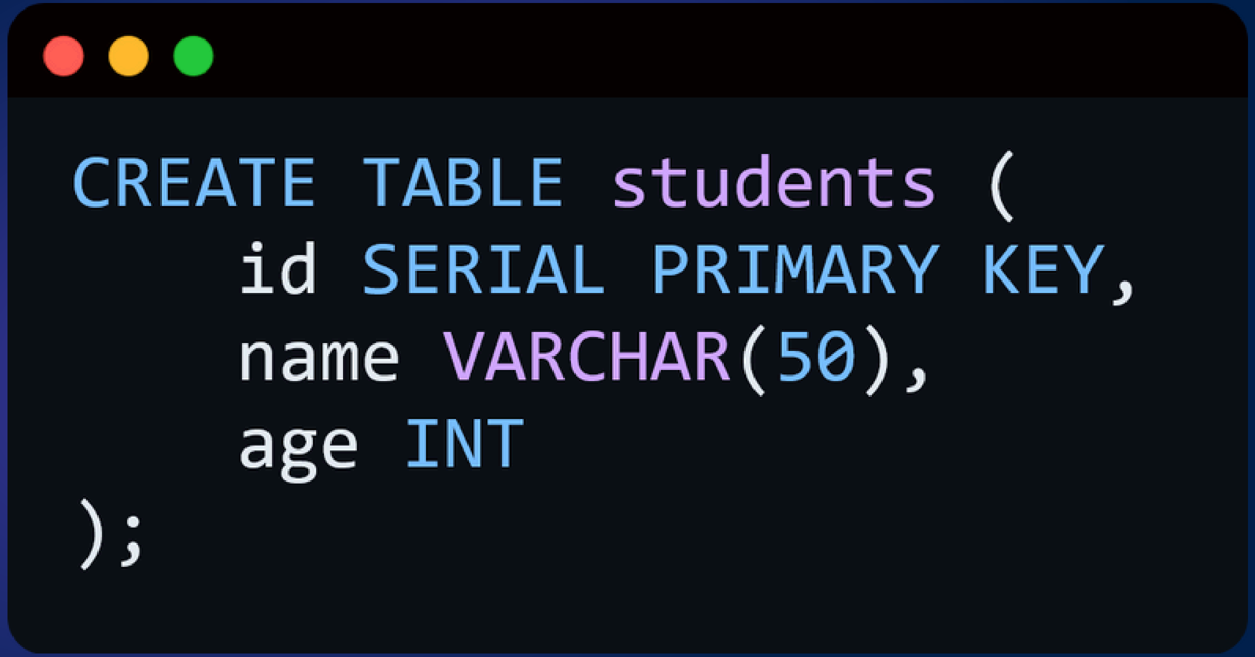
```sql
INSERT INTO students
VALUES (1, 'Sadia', 22);
```



```sql
INSERT INTO students
VALUES ('Sadia', 22);    -- ❌ error
```



```sql
INSERT INTO students (name, age)
VALUES ('Sadia', 22);
```



```sql
CREATE TABLE students (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INT
);
```