PostgreSQL Constraints

Agenda

- Types of Constraints
- Defining and Modifying Constraints
- Best Practices

Types of Constraints

- **CHECK**
- **DEFAULT Values**
- NOT NULL
- UNIQUE
- PRIMARY KEY,

  COMPOSITE

  PRIMARY KEY
- FOREIGN KEY

Constraints

- are used to define rules for columns in a database table. They ensure that no invalid data is entered into the database.

- gives as much control over the data in tables as you wish. If a user attempts to store data in a column that would violate a constraint, an error is raised.

Constraints

- Rules applied to table columns
- Enforce data integrity and accuracy
- Prevent invalid or inconsistent data
- Maintain reliable database structures

CHECK

The CHECK constraint ensures that the data in a column meets a particular condition.

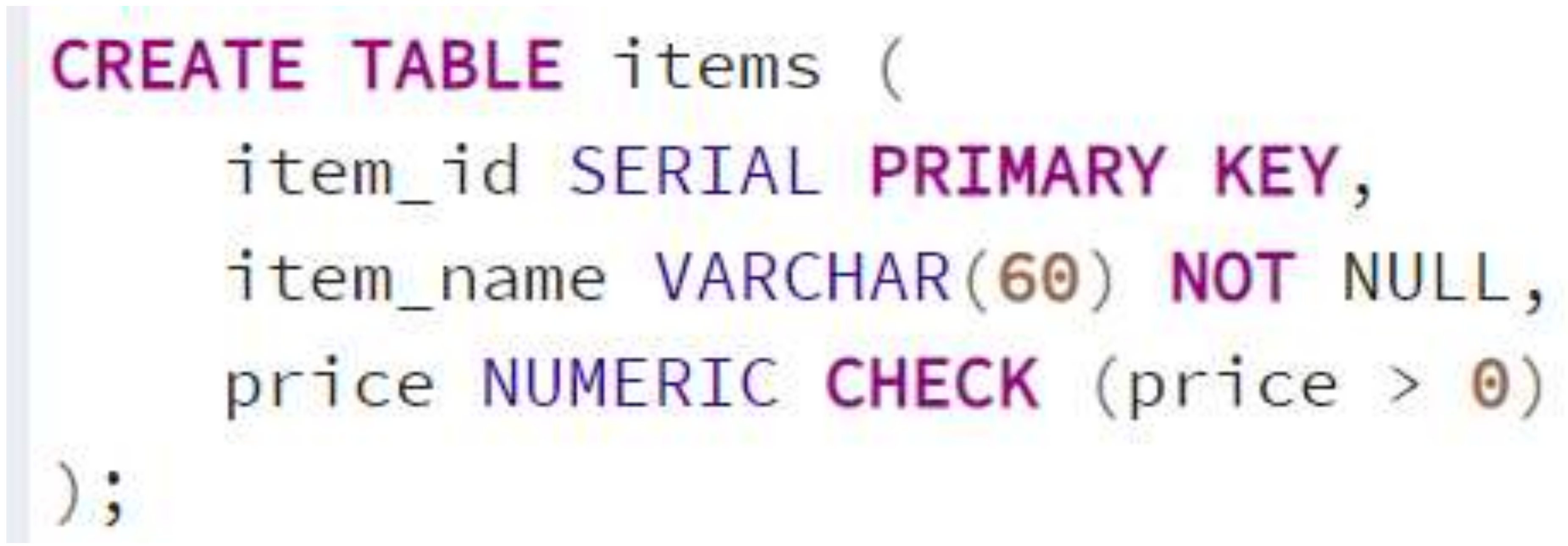
With CHECK constraint, it is possible to define rules such as value ranges, specific formats, or logical relationships between columns.

Syntax:

variable_name Data-type CHECK(condition);

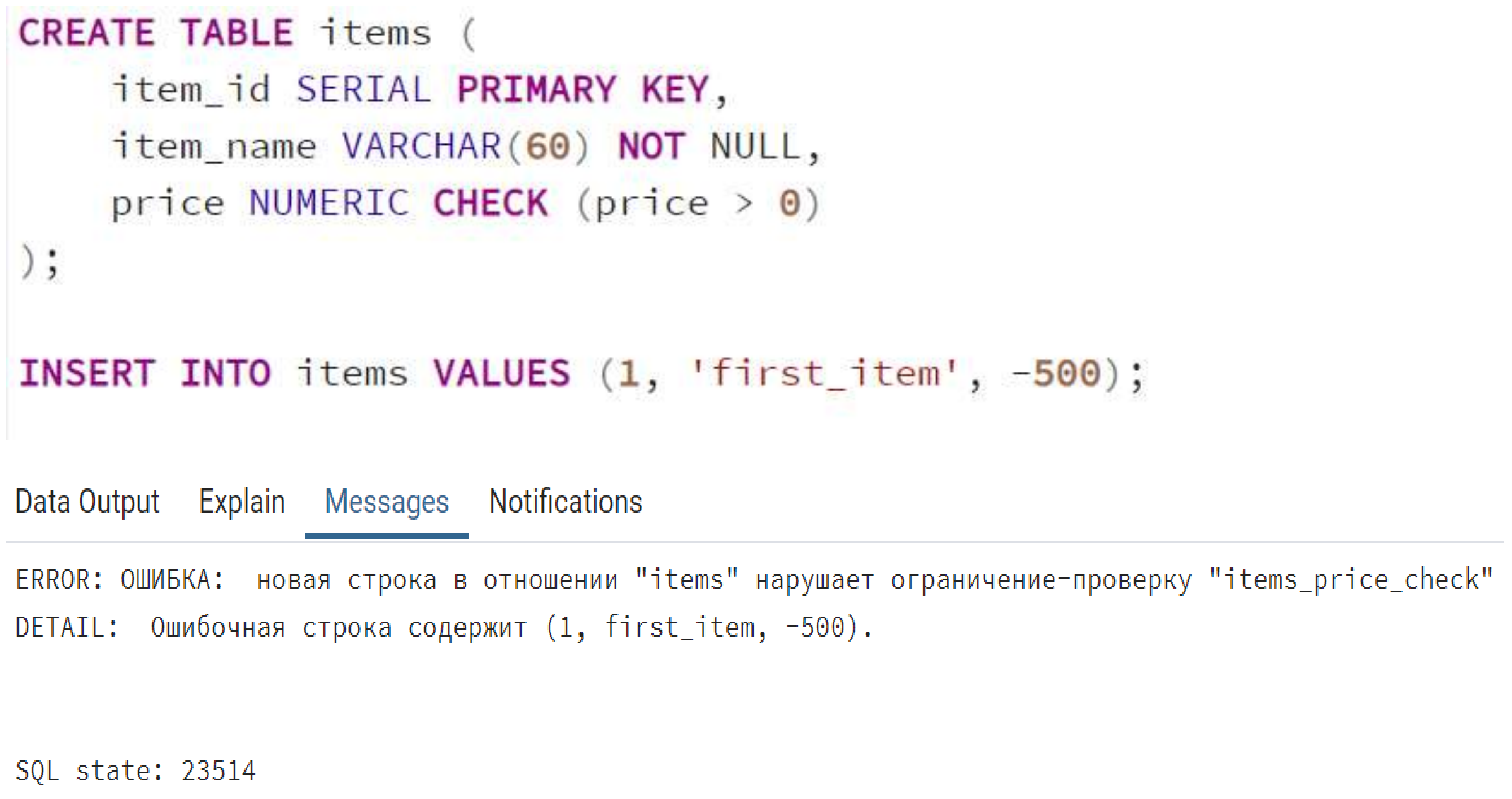
CHECK

- Specifies a condition that must be true for each row
- Enforces domain integrity



```
CREATE TABLE items (
    item_id SERIAL PRIMARY KEY,
    item_name VARCHAR(60) NOT NULL,
    price NUMERIC CHECK (price > 0)
);
```

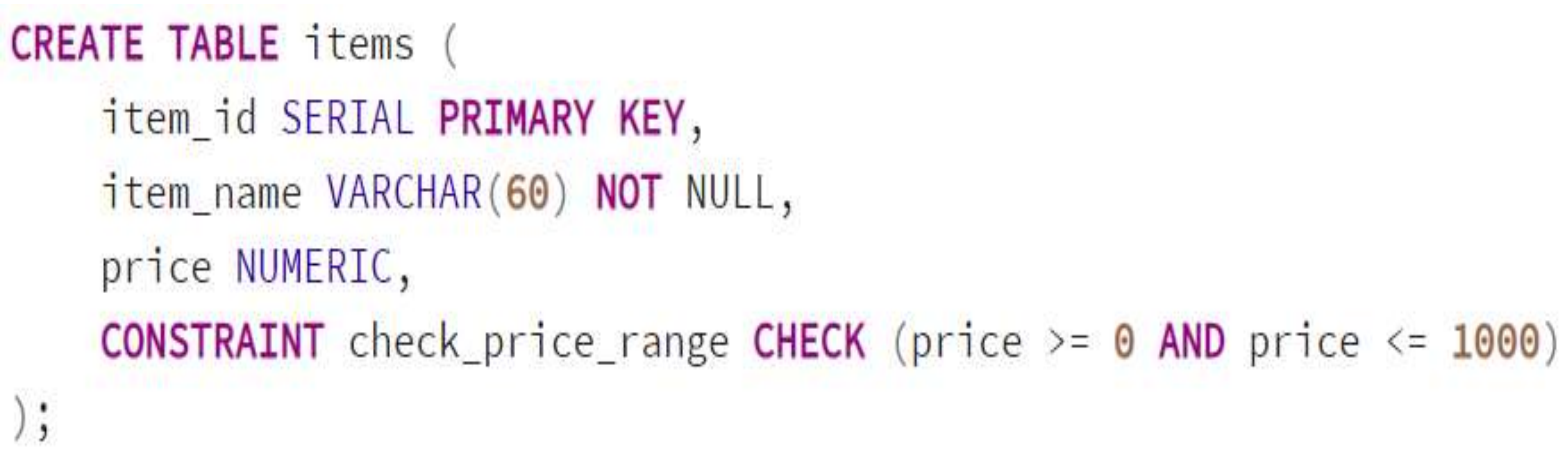
CHECK



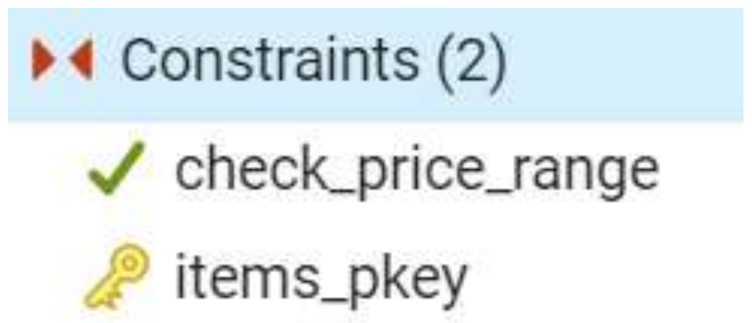
```
CREATE TABLE items (
    item_id SERIAL PRIMARY KEY,
    item_name VARCHAR(60) NOT NULL,
    price NUMERIC CHECK (price > 0)
);

INSERT INTO items VALUES (1, 'first_item', -500);
```

```
ERROR: ОШИБКА:  новая строка в отношении "items" нарушает ограничение-проверку "items_price_check"
DETAIL:  Ошибочная строка содержит (1, first_item, -500).


SQL state: 23514
```

Numeric Range

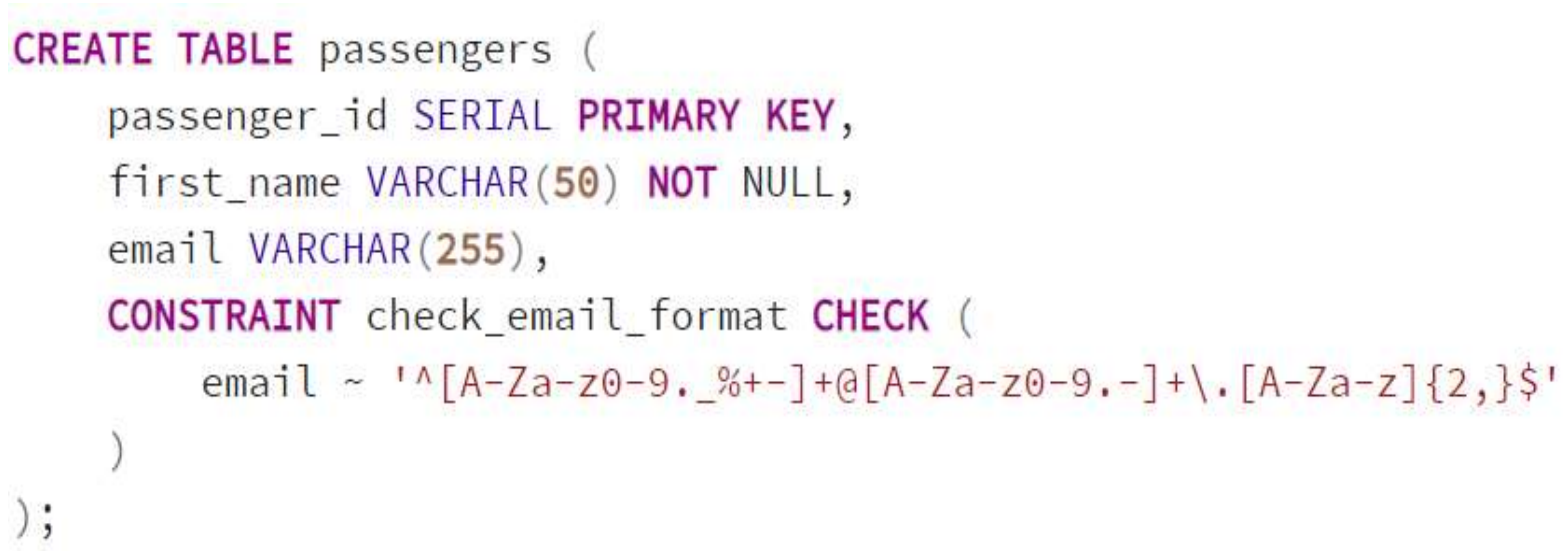


```
CREATE TABLE items (
    item_id SERIAL PRIMARY KEY,
    item_name VARCHAR(60) NOT NULL,
    price NUMERIC,
    CONSTRAINT check_price_range CHECK (price >= 0 AND price <= 1000)
);
```



```
Constraints (2)
  ✓ check_price_range
    items_pkey
```

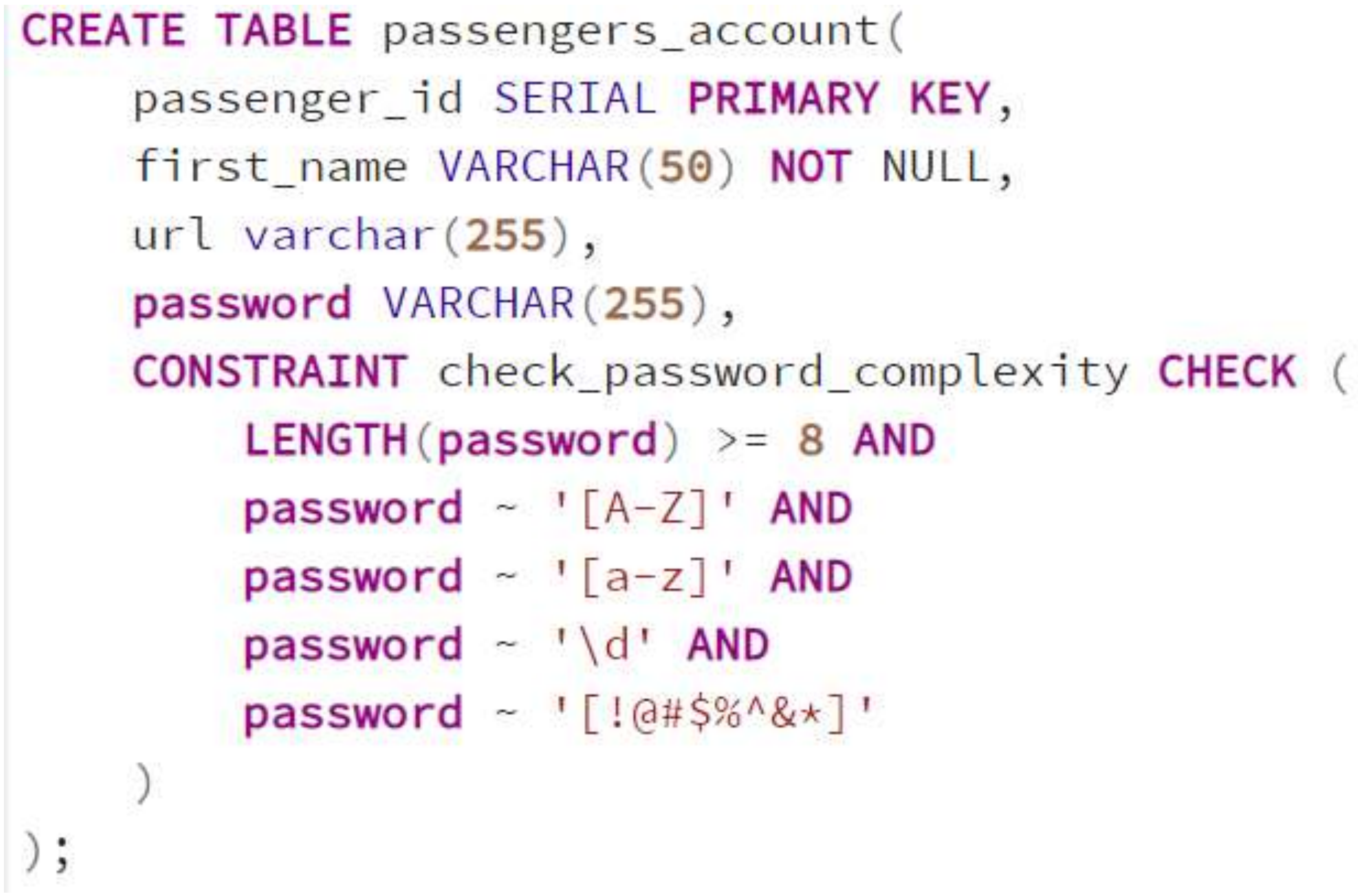
String Patterns



```
CREATE TABLE passengers (
    passenger_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    email VARCHAR(255),
    CONSTRAINT check_email_format CHECK (
        email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'
    )
);
```

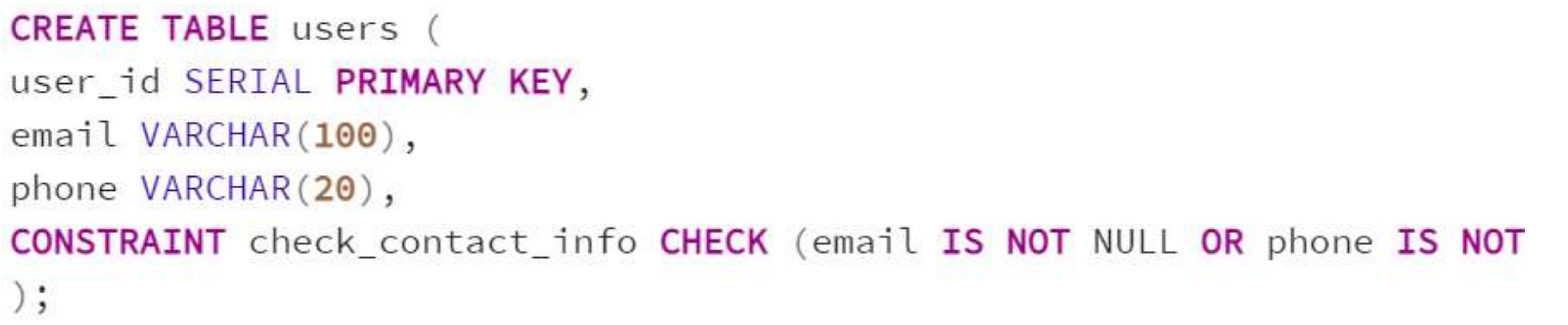
^[A-Za-z0-9._%+-]+:containing allowed characters.
@: symbol separating local part and domain.
[A-Za-z0-9.-]+: Domain name with allowed characters.
[A-Za-z]{2,}$: Top-level domain with at least two letters.

Multiple Pattern Constraints



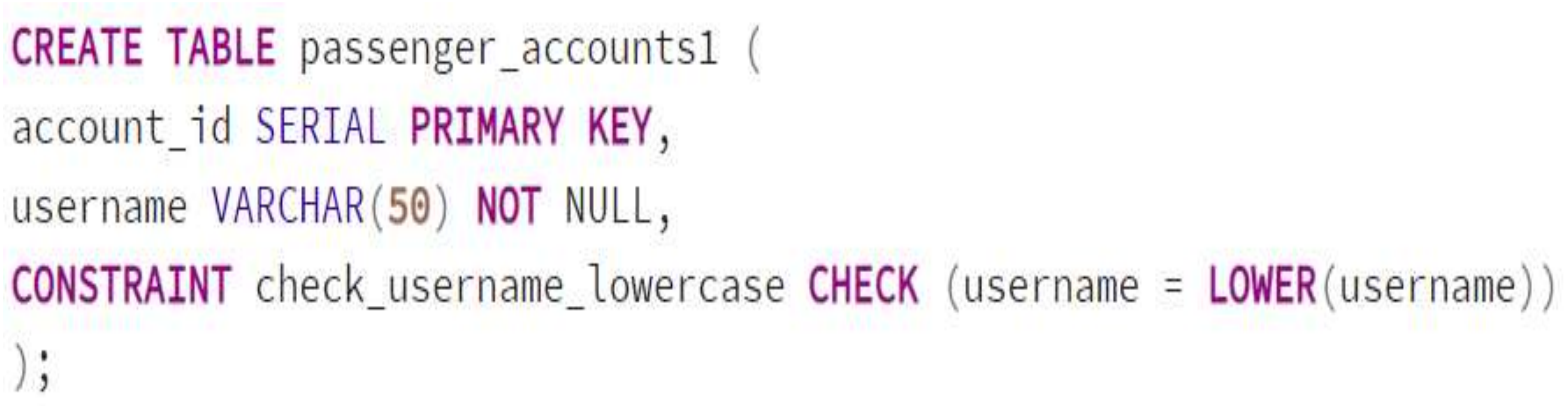
```sql
CREATE TABLE passengers_account(
    passenger_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    url varchar(255),
    password VARCHAR(255),
    CONSTRAINT check_password_complexity CHECK (
        LENGTH(password) >= 8 AND
        password ~ '[A-Z]' AND
        password ~ '[a-z]' AND
        password ~ '\d' AND
        password ~ '[!@#$%^&*]'
    )
);
```

Multiple Column Constraints



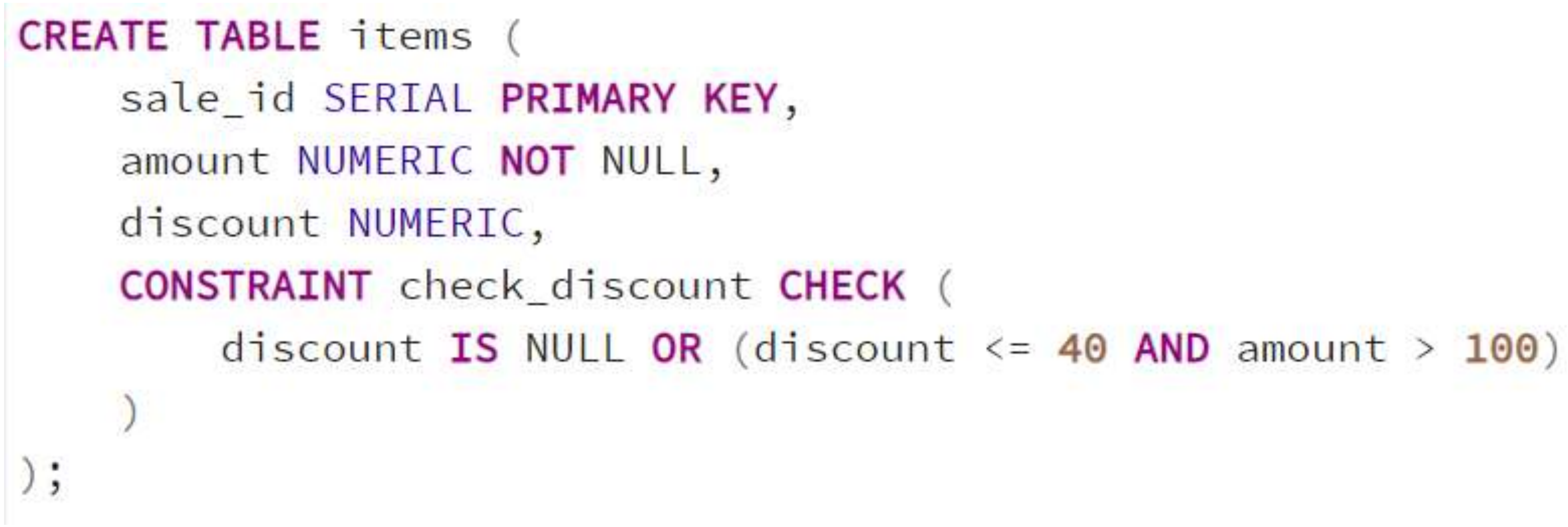
```
CREATE TABLE users (
user_id SERIAL PRIMARY KEY,
email VARCHAR(100),
phone VARCHAR(20),
CONSTRAINT check_contact_info CHECK (email IS NOT NULL OR phone IS NOT
);
```

Functions in Constraints



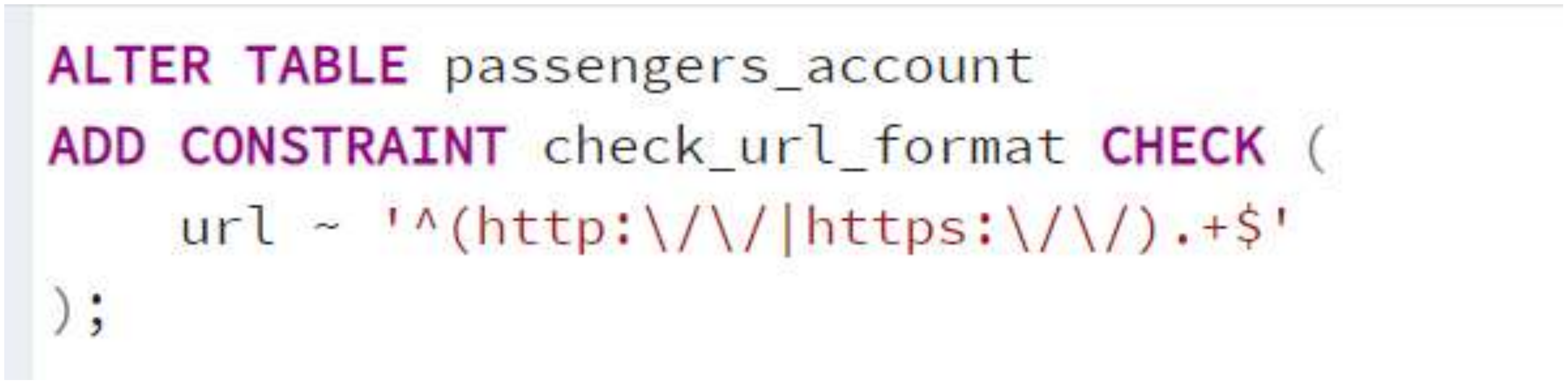
```
CREATE TABLE passenger_accounts1 (
account_id SERIAL PRIMARY KEY,
username VARCHAR(50) NOT NULL,
CONSTRAINT check_username_lowercase CHECK (username = LOWER(username))
);
```

Composite Constraints with Multiple Conditions



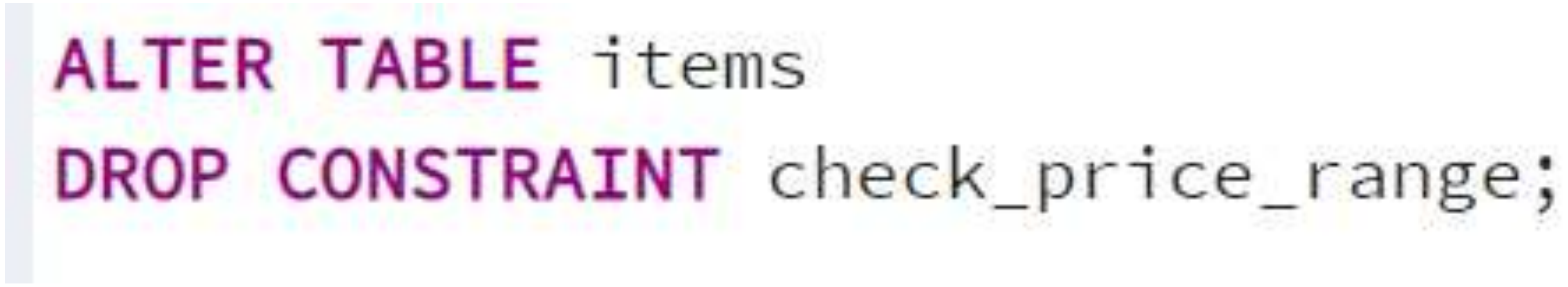
```sql
CREATE TABLE items (
    sale_id SERIAL PRIMARY KEY,
    amount NUMERIC NOT NULL,
    discount NUMERIC,
    CONSTRAINT check_discount CHECK (
        discount IS NULL OR (discount <= 40 AND amount > 100)
    )
);
```

Updating existing tables with CHECK



```
ALTER TABLE passengers_account
ADD CONSTRAINT check_url_format CHECK (
    url ~ '^(http:\/\/|https:\/\/).+$'
);
```

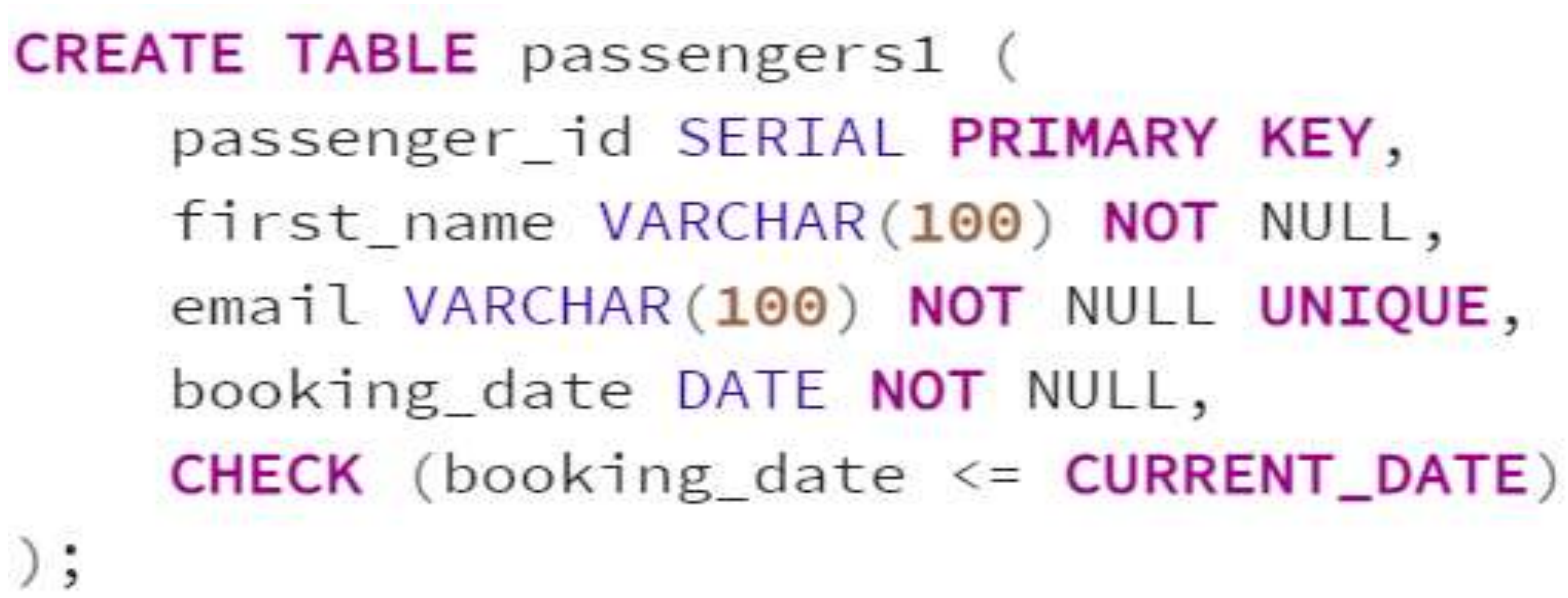
Removing a CHECK Constraint



```
ALTER TABLE items
DROP CONSTRAINT check_price_range;
```

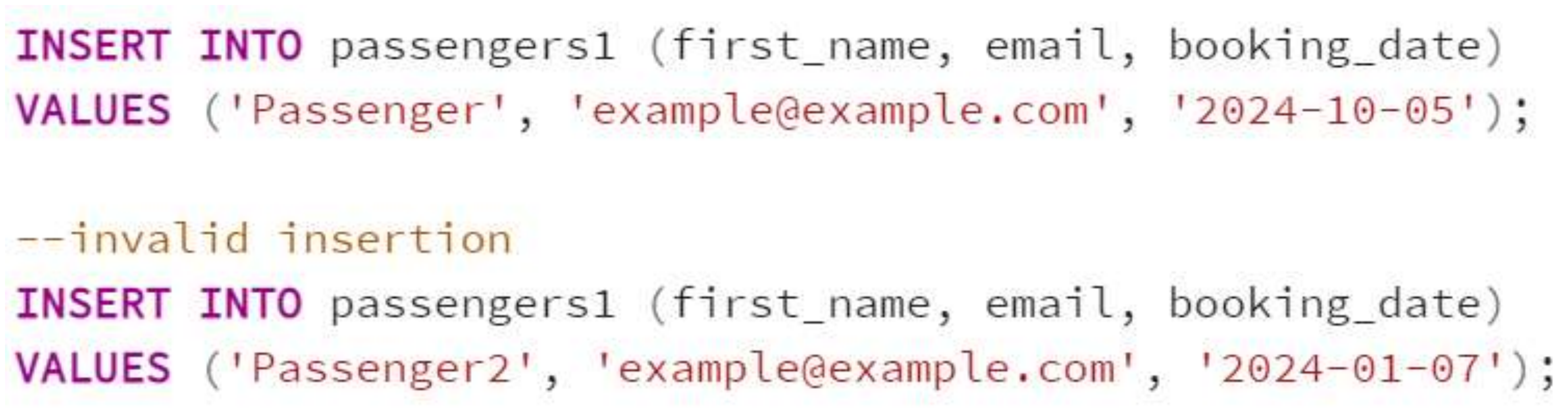
Multiple constraints

In PostgreSQL, it is possible to add multiple constraints to a table

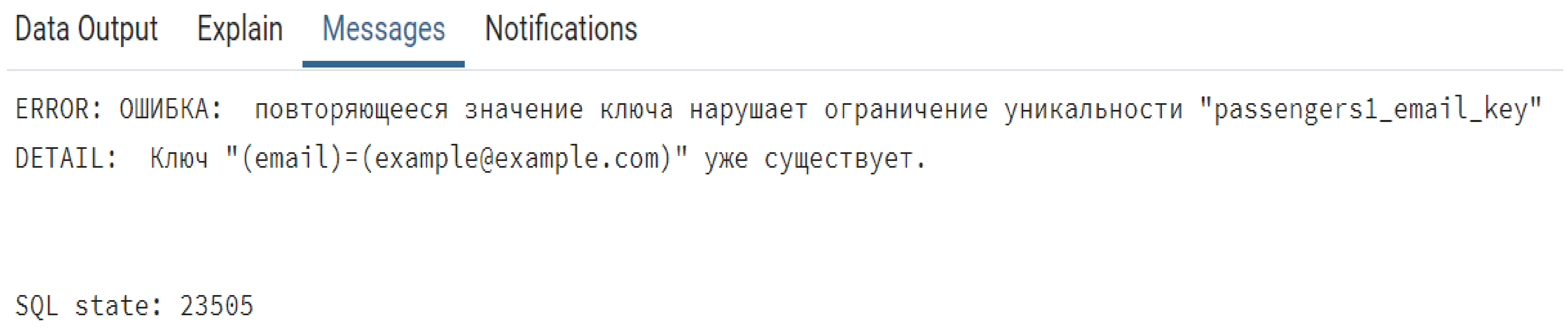


```
CREATE TABLE passengers1 (
    passenger_id SERIAL PRIMARY KEY,
    first_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    booking_date DATE NOT NULL,
    CHECK (booking_date <= CURRENT_DATE)
);
```

Multiple constraints



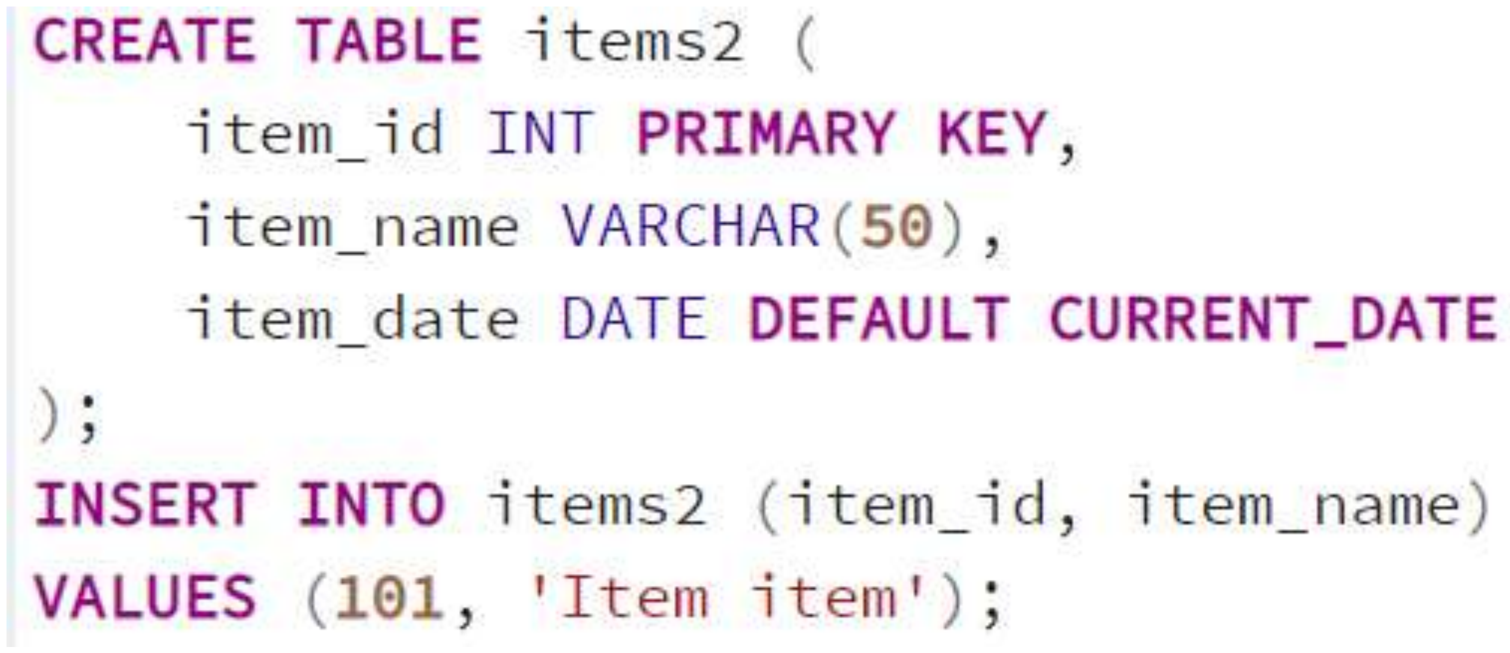
```sql
INSERT INTO passengers1 (first_name, email, booking_date)
VALUES ('Passenger', 'example@example.com', '2024-10-05');

--invalid insertion
INSERT INTO passengers1 (first_name, email, booking_date)
VALUES ('Passenger2', 'example@example.com', '2024-01-07');
```



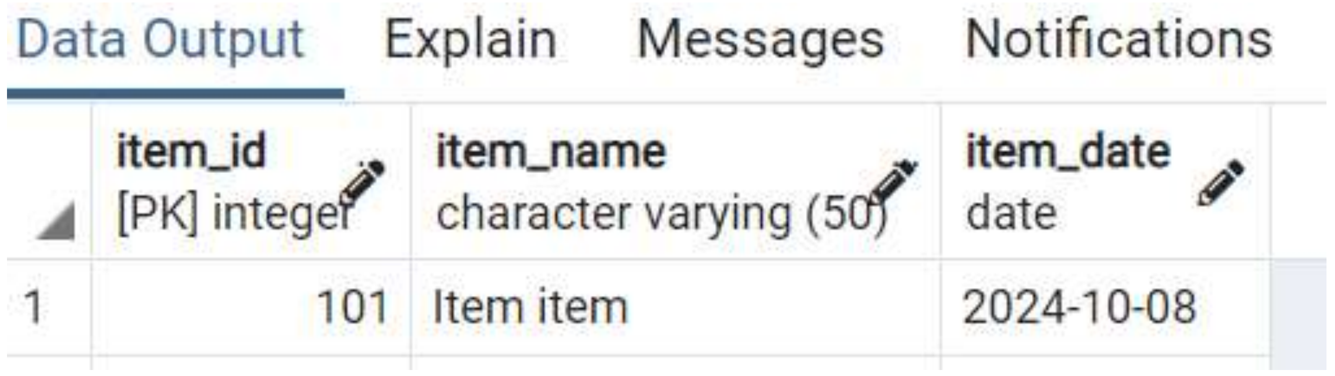
```
ERROR: ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "passengers1_email_key"
DETAIL:  Ключ "(email)=(example@example.com)" уже существует.


SQL state: 23505
```

DEFAULT constraint

The **DEFAULT** constraint is used to specify a default value for a column. If no value is specified when inserting a new row, the default value will automatically be used.



```
CREATE TABLE items2 (
    item_id INT PRIMARY KEY,
    item_name VARCHAR(50),
    item_date DATE DEFAULT CURRENT_DATE
);
INSERT INTO items2 (item_id, item_name)
VALUES (101, 'Item item');
```



Data Output Explain Messages Notifications

	item_id [PK] integer	item_name character varying (50)	item_date date
1	101	Item item	2024-10-08

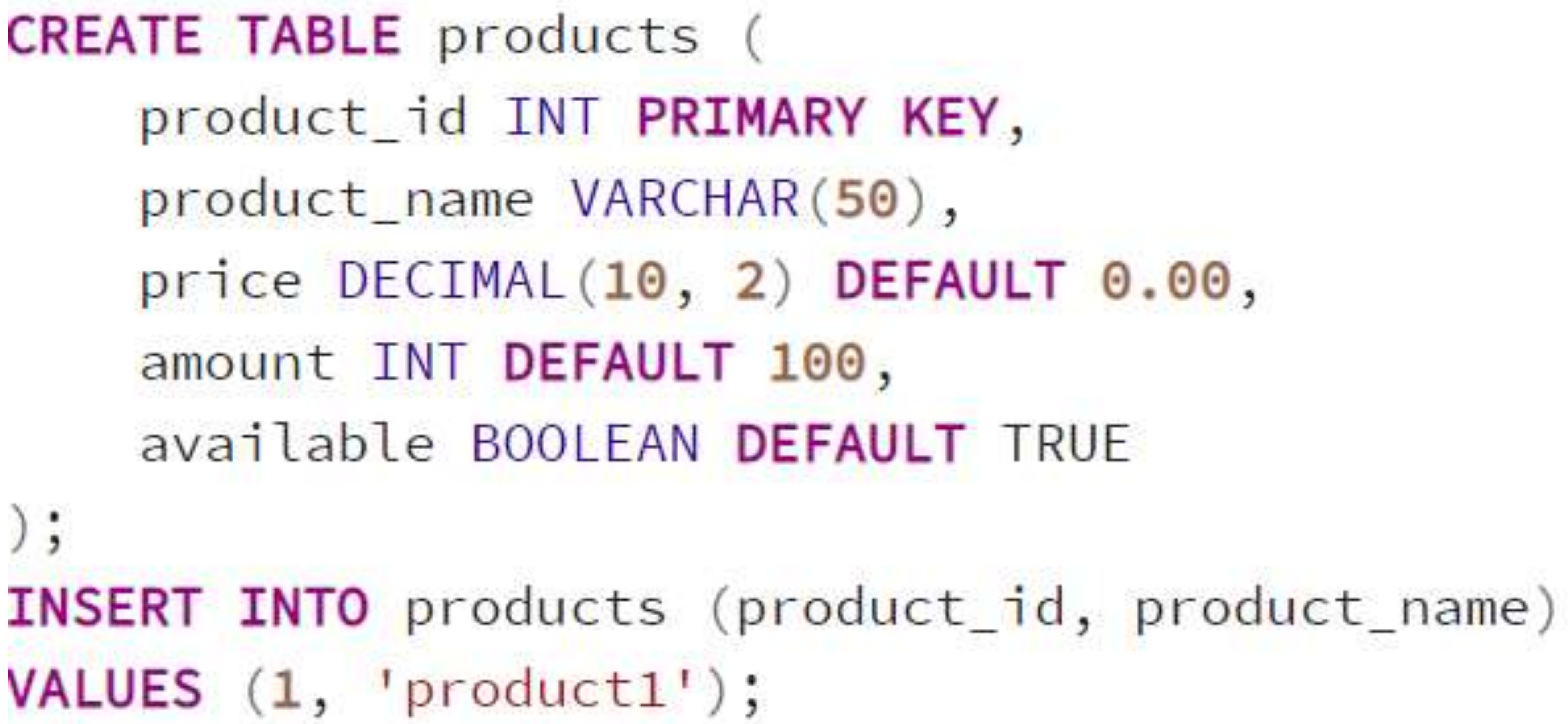
DEFAULT Value Constraint

- **Automatic Value Assignment:** If a value for a column with a DEFAULT constraint is not provided during insertion, the database assigns the default value.

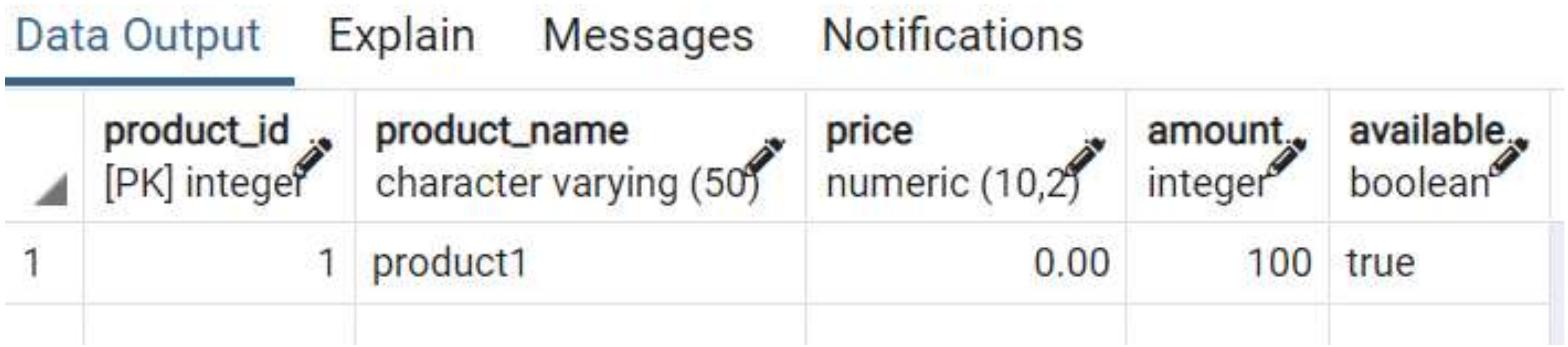
- **Optional Use:** The DEFAULT constraint is optional. If not specified, the column will have a value of NULL if it allows nulls, or it must have a value if it does not allow nulls.

- **Flexible Types:** The default value can be a constant value, an expression, or a function, depending on the data type of the column.

DEFAULT constraint



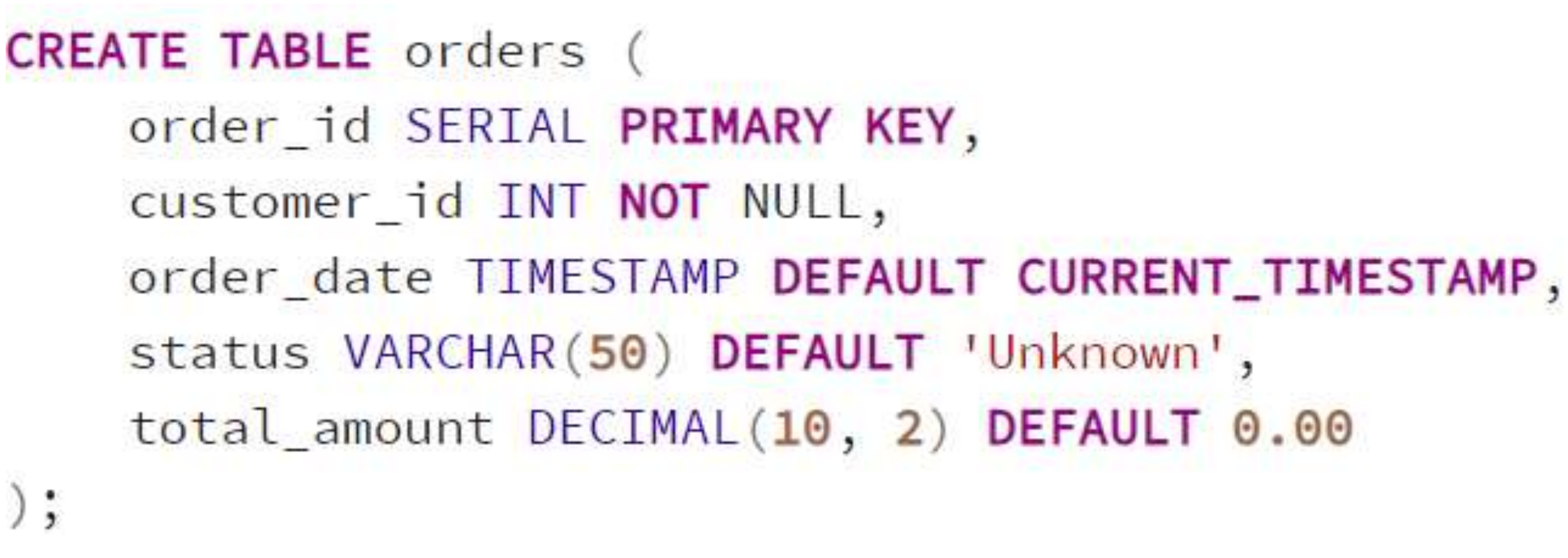
```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50),
    price DECIMAL(10, 2) DEFAULT 0.00,
    amount INT DEFAULT 100,
    available BOOLEAN DEFAULT TRUE
);
INSERT INTO products (product_id, product_name)
VALUES (1, 'product1');
```



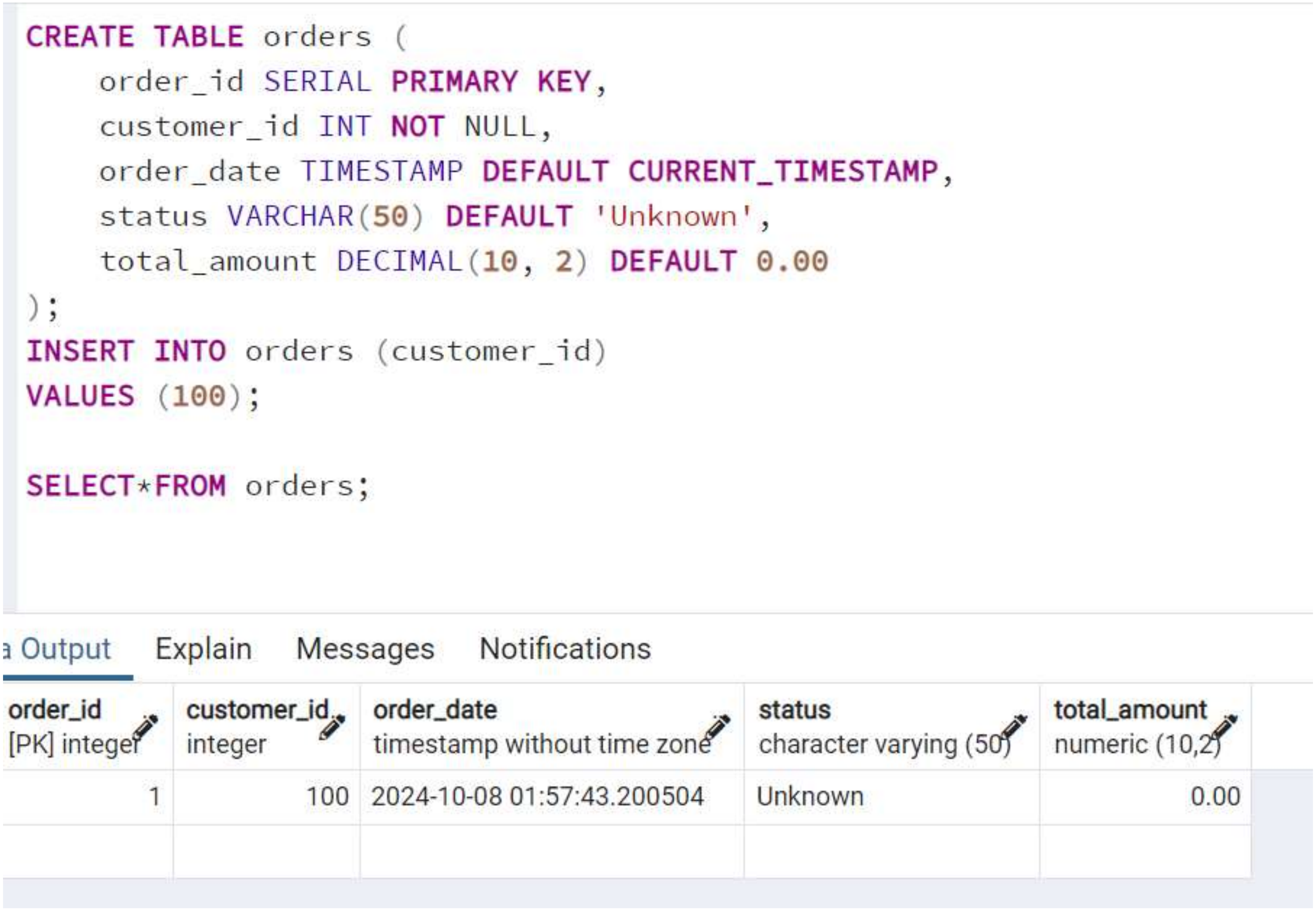
	product_id [PK] integer	product_name character varying (50)	price numeric (10,2)	amount integer	available boolean
1	1	product1	0.00	100	true

DEFAULT Value Constraint

The **DEFAULT value** constraint in SQL is used to specify a default value for a column in a table.
When a new row is inserted into the table without specifying a value for that column, the default value will be automatically assigned.



```
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50) DEFAULT 'Unknown',
    total_amount DECIMAL(10, 2) DEFAULT 0.00
);
```

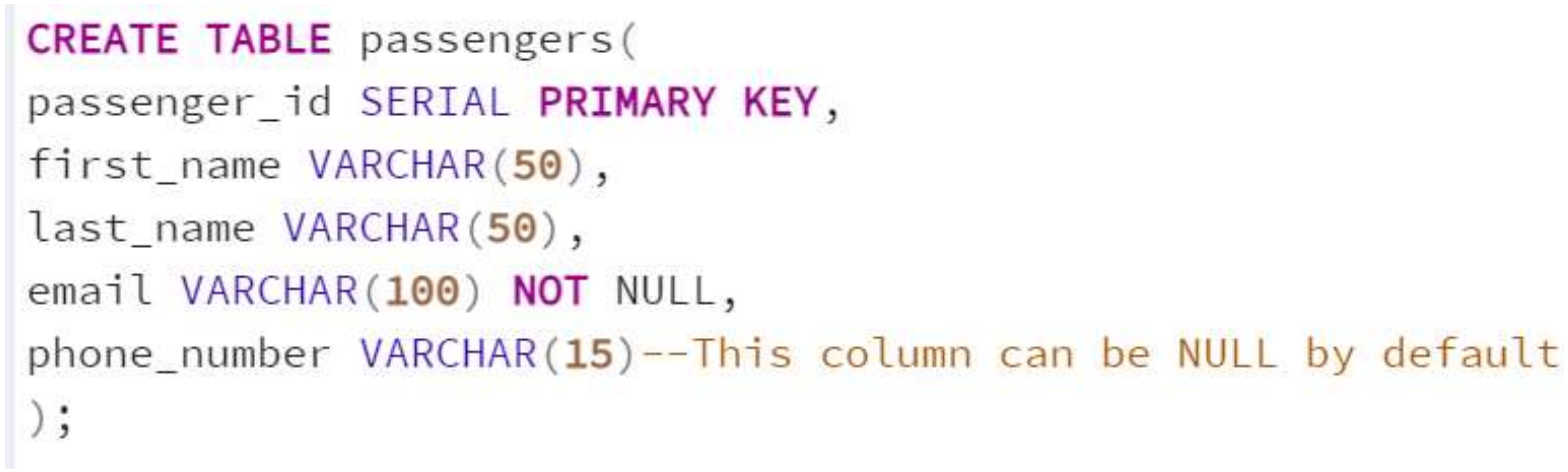


```
CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(50) DEFAULT 'Unknown',
    total_amount DECIMAL(10, 2) DEFAULT 0.00
);
INSERT INTO orders (customer_id)
VALUES (100);

SELECT*FROM orders;
```

order_id [PK] integer	customer_id integer	order_date timestamp without time zone	status character varying (50)	total_amount numeric (10,2)
1	100	2024-10-08 01:57:43.200504	Unknown	0.00

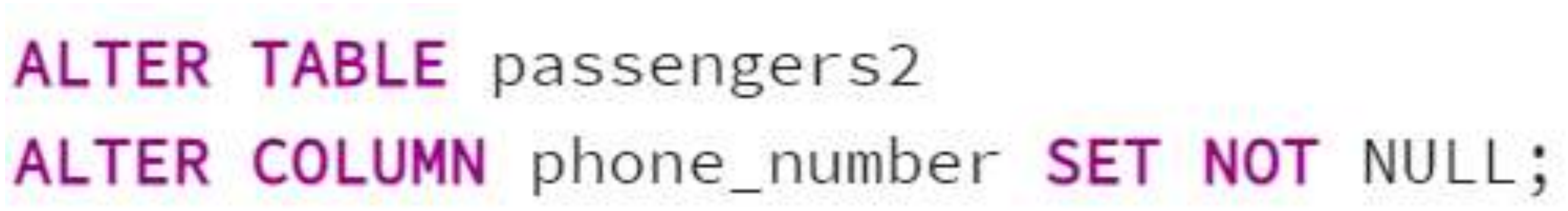
NULL and NOT NULL Constraints

- A column with a NULL constraint can accept NULL values.
- By default, columns in PostgreSQL allow NULL values unless defined with the NOT NULL constraint.
- NULL: Represents the absence of a value. It's different from an empty string or zero.
- NOT NULL: Requires that a value is always present in the column.



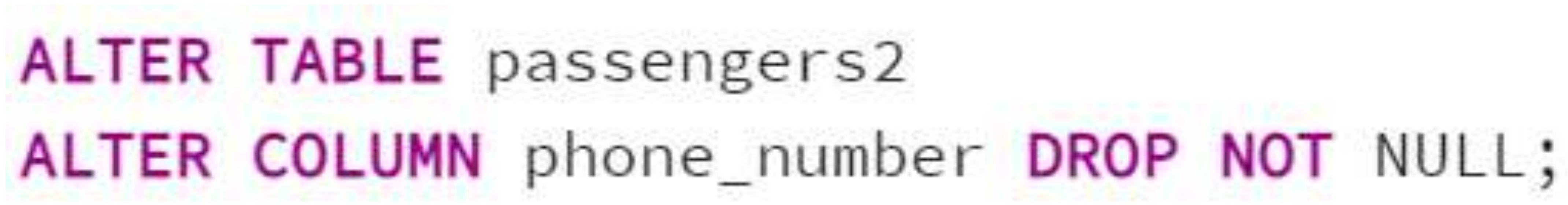
```sql
CREATE TABLE passengers(
passenger_id SERIAL PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
email VARCHAR(100) NOT NULL,
phone_number VARCHAR(15)--This column can be NULL by default
);
```

Alter Table to Add NOT NULL Constraints



```sql
ALTER TABLE passengers2
ALTER COLUMN phone_number SET NOT NULL;
```

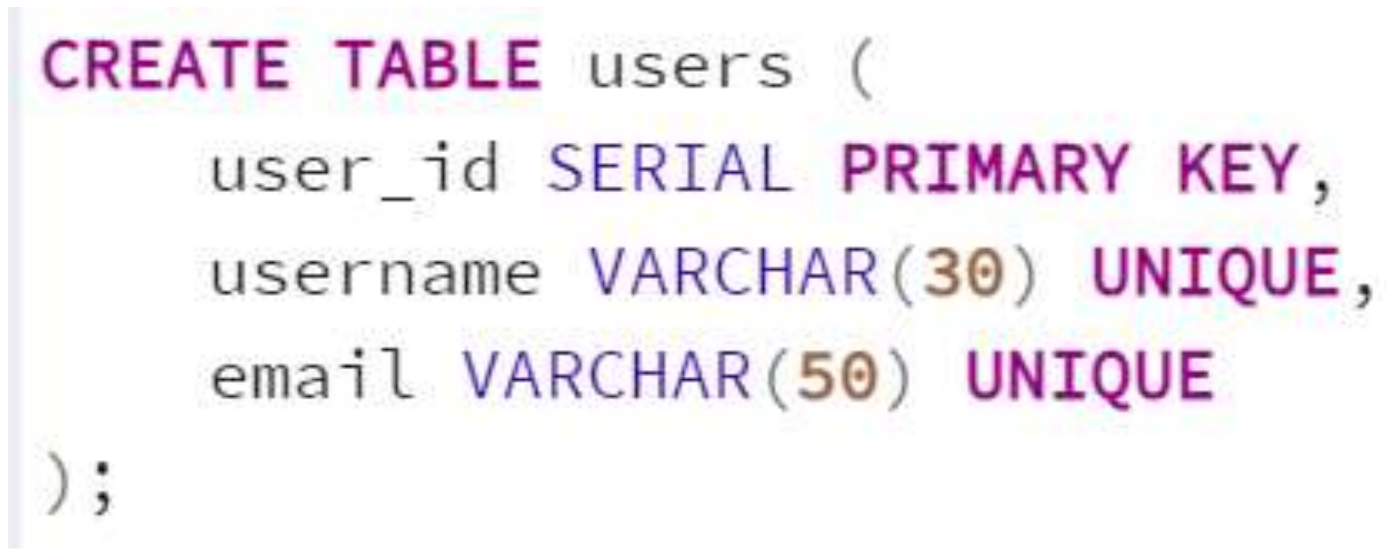
Delete NOT NULL Constraints



```sql
ALTER TABLE passengers2
ALTER COLUMN phone_number DROP NOT NULL;
```

Unique

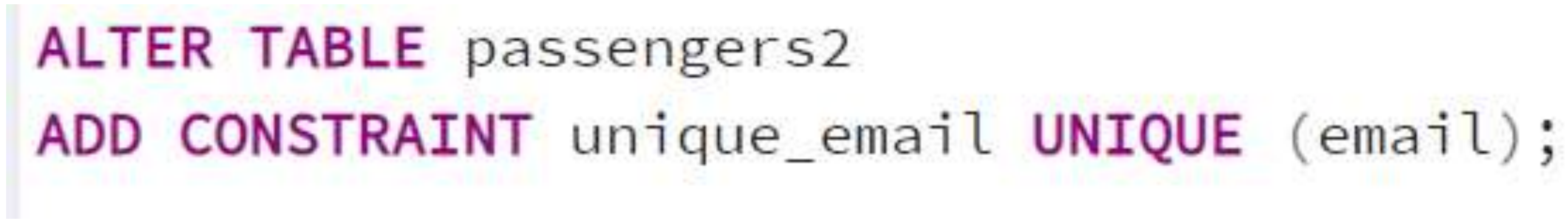
- A UNIQUE constraint in SQL ensures that all the values in a column or a group of columns are distinct



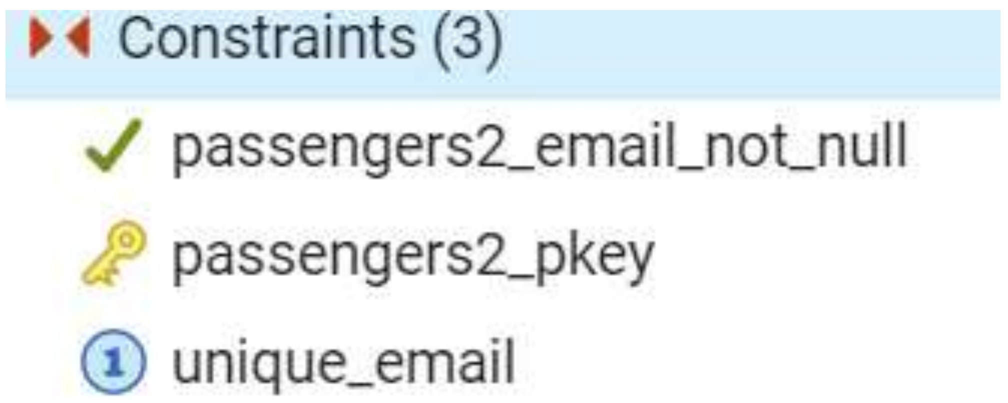
```sql
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(30) UNIQUE,
    email VARCHAR(50) UNIQUE
);
```

Unique

Adding a UNIQUE constraint to an existing table:



```
ALTER TABLE passengers2
ADD CONSTRAINT unique_email UNIQUE (email);
```



Constraints (3)
- passengers2_email_not_null
- passengers2_pkey
- unique_email

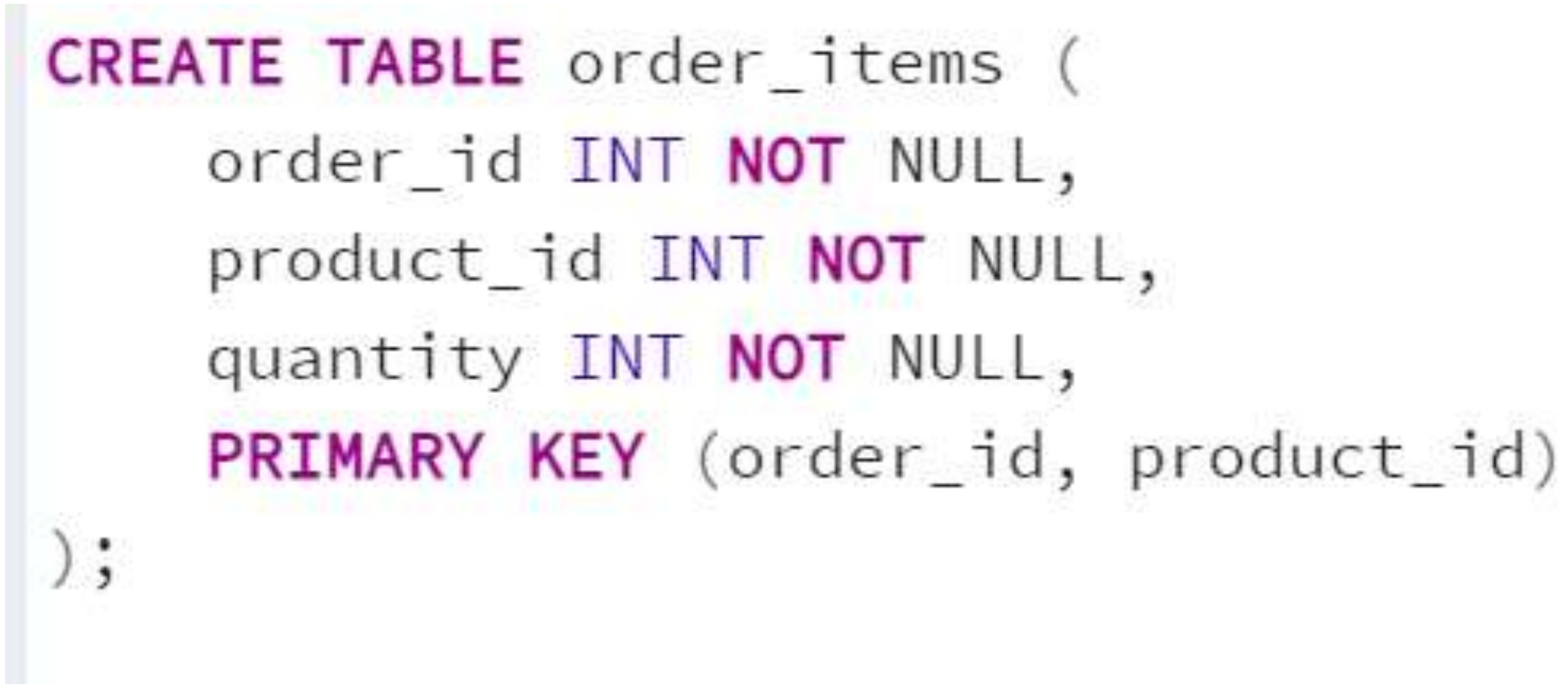
Unique

- The UNIQUE constraint allows NULL values. However, each NULL is treated as distinct, meaning multiple NULL values are allowed in a column with a UNIQUE constraint.

- Unlike the PRIMARY KEY constraint, which also enforces uniqueness, a table can have multiple UNIQUE constraints.

Composite Primary key

A Composite Primary Key is a primary key that spans more than one column in a table. It is used to uniquely identify records based on a combination of columns, ensuring that no two rows have the same combination of values in those columns.

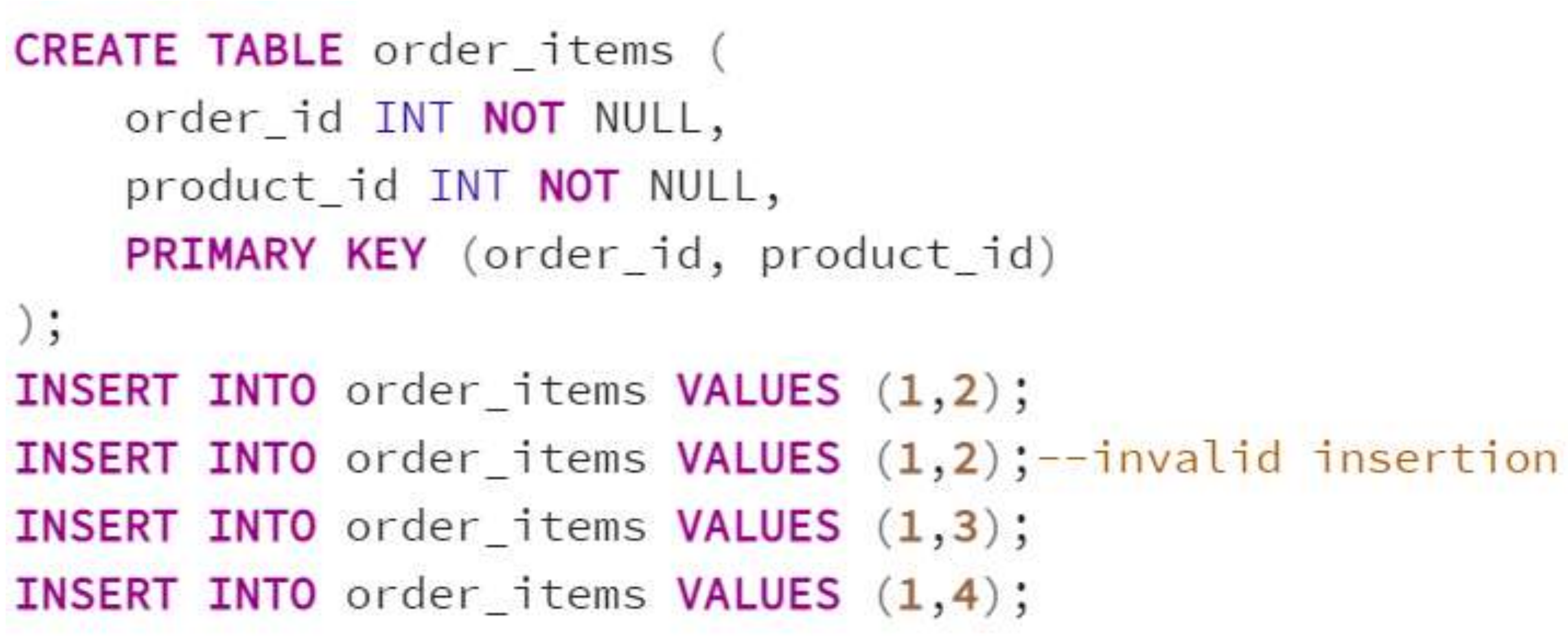


```sql
CREATE TABLE order_items (
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    PRIMARY KEY (order_id, product_id)
);
```

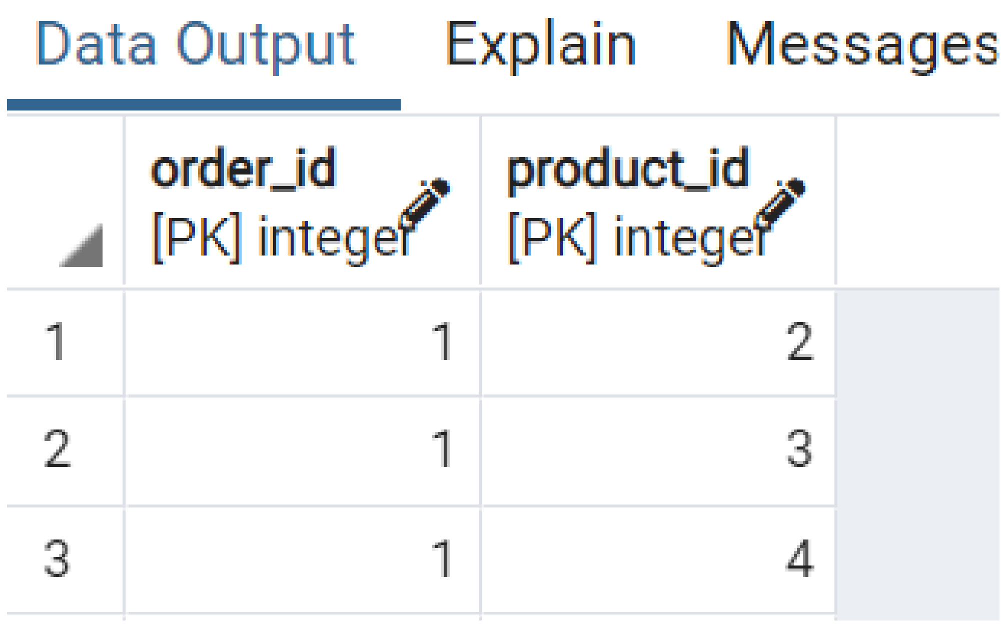
Unique Key vs. Composite Primary Key

Feature	Unique Key	Composite Primary Key
Uniqueness	Enforces uniqueness for a single or multiple columns.	Enforces uniqueness based on a combination of multiple columns.
NULL Values	Allows `NULL` values, but `NULL` values are treated as unique.	Does **not** allow `NULL` values in any of the columns.
Number per Table	A table can have multiple unique keys.	A table can have only one composite primary key.
Use Case	Ensures uniqueness of a specific attribute (e.g., email, username).	Uniquely identifies records where a single column isn't sufficient.
Relationship	Does not imply any referential relationship with other tables.	Often used in complex relationships (e.g., many-to-many) where a composite of columns is needed for uniqueness.
Index	Automatically creates a unique index.	Automatically creates a unique composite index.
Constraint Name	Can be named explicitly.	Typically named by the database, but can also be explicitly named.

Composite Primary key



```
CREATE TABLE order_items (
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    PRIMARY KEY (order_id, product_id)
);
INSERT INTO order_items VALUES (1,2);
INSERT INTO order_items VALUES (1,2);--invalid insertion
INSERT INTO order_items VALUES (1,3);
INSERT INTO order_items VALUES (1,4);
```



	order_id [PK] integer	product_id [PK] integer
1	1	2
2	1	3
3	1	4

Foreign keys

A foreign key establishes a relationship between two tables by referencing the primary key of another table.

Foreign keys

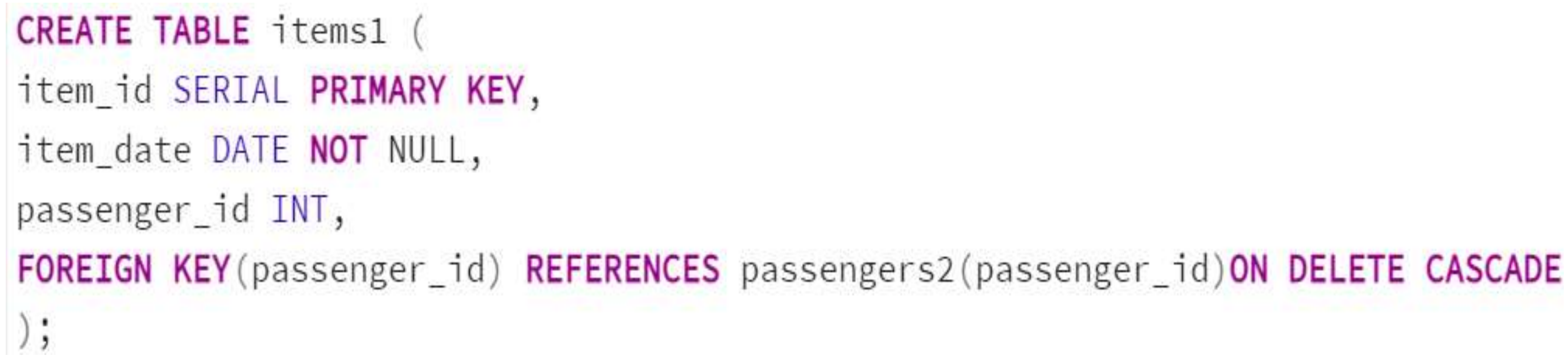
- **Ensures Referential Integrity:** Prevents inserting data in the child table that doesn't exist in the parent table.

- **Links Tables:** Creates a relationship between the child and parent tables.

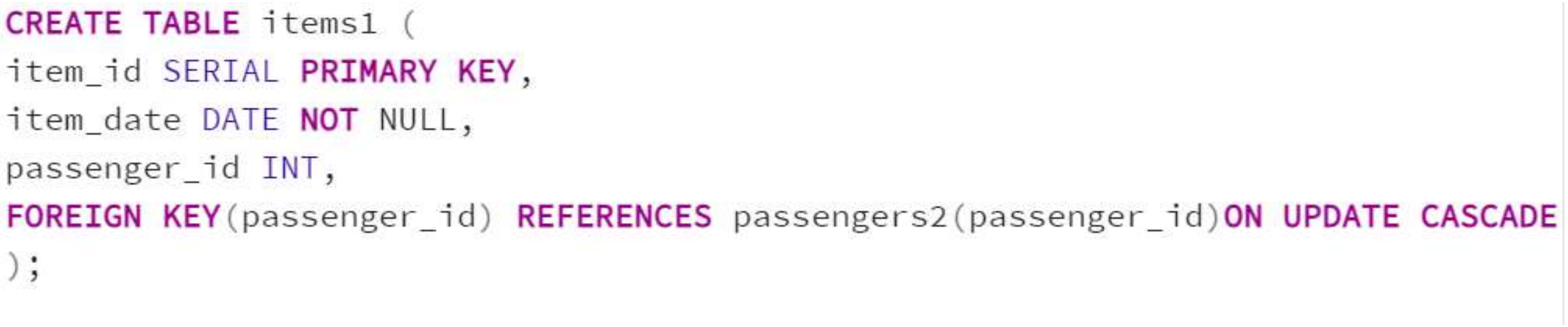
- **Cascade Options:** Offers options like **ON DELETE CASCADE or ON UPDATE CASCADE.**

ON DELETE and ON UPDATE

- **ON DELETE CASCADE:** Deletes the related rows in the child table when a row in the parent table is deleted.
- **ON UPDATE CASCADE**: Automatically updates the foreign key in the child table when the referenced row in the parent table is updated.
- **ON DELETE SET NULL**: Sets the foreign key to NULL in the child table when a referenced row in the parent table is deleted.
- **ON DELETE RESTRICT**: Prevents the deletion of a row in the parent table if it is referenced by rows in the child table.



```
CREATE TABLE items1 (
item_id SERIAL PRIMARY KEY,
item_date DATE NOT NULL,
passenger_id INT,
FOREIGN KEY(passenger_id) REFERENCES passengers2(passenger_id)ON DELETE CASCADE
);
```



```
CREATE TABLE items1 (
item_id SERIAL PRIMARY KEY,
item_date DATE NOT NULL,
passenger_id INT,
FOREIGN KEY(passenger_id) REFERENCES passengers2(passenger_id)ON UPDATE CASCADE
);
```