Here's a simple example of a table in PostgreSQL, complete with DDL statements, data insertion (including NULL values), and queries that demonstrate how to deal with NULL values using various SQL keywords.

**Sample Table: products**

1. **Creating the Table**

sql

Copy code

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price NUMERIC(10, 2),

quantity INT,

description TEXT

);

**Inserting Data (Including NULL Values)**

1. **Inserting Data into the Table**

sql

Copy code

INSERT INTO products (product\_name, price, quantity, description) VALUES

('Laptop', 999.99, 50, 'High performance laptop'),

('Smartphone', NULL, 100, 'Latest model smartphone'), -- price is NULL

('Tablet', 299.99, NULL, NULL), -- quantity and description are NULL

('Smartwatch', 199.99, 75, NULL); -- description is NULL

**Dealing with NULL Values**

1. **Queries Involving NULL Values**

* **Selecting All Products**

sql

Copy code

SELECT \* FROM products;

* **Filtering Out NULL Values**

To retrieve only products with a non-NULL price:

sql

Copy code

SELECT \* FROM products WHERE price IS NOT NULL;

* **Finding Products with NULL Values**

To retrieve products where the quantity is NULL:

sql

Copy code

SELECT \* FROM products WHERE quantity IS NULL;

* **Using COALESCE**

To replace NULL values with a default value in the results (e.g., displaying "N/A" for NULL descriptions):

sql

Copy code

SELECT product\_name, COALESCE(description, 'N/A') AS description FROM products;

* **Counting NULL Values**

To count how many products have a NULL price:

sql

Copy code

SELECT COUNT(\*) AS null\_price\_count FROM products WHERE price IS NULL;

* **Using CASE Statement**

To categorize products based on their availability (using NULL values):

sql

Copy code

SELECT product\_name,

CASE

WHEN quantity IS NULL THEN 'Unknown'

WHEN quantity > 0 THEN 'In Stock'

ELSE 'Out of Stock'

END AS availability

FROM products;

**Summary of Keywords and Queries**

* **IS NULL**: Checks for NULL values.
* **IS NOT NULL**: Checks for non-NULL values.
* **COALESCE**: Returns the first non-NULL value in the list of arguments.
* **COUNT(\*)**: Counts rows, including those with NULL values (you can filter with WHERE for specific counts).
* **CASE**: Allows conditional logic to categorize or transform the data in the result set.

This example demonstrates how to create a table, insert data including NULL values, and perform various queries to handle NULL values effectively.