Step 1: Create Tables

We'll create two tables: products and price_ranges.

1. **products** table:

   product_id: INT (Primary Key)

   product_name: VARCHAR(50)

   price: DECIMAL(10, 2)

price_ranges table:

   range_id: INT (Primary Key)

   range_name: VARCHAR(50)

   min_price: DECIMAL(10, 2)

   max_price: DECIMAL(10, 2)

```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50),
    price DECIMAL(10, 2)
);

CREATE TABLE price_ranges (
    range_id INT PRIMARY KEY,
    range_name VARCHAR(50),
    min_price DECIMAL(10, 2),
    max_price DECIMAL(10, 2)
);
```

Step 2: Insert Records

We'll insert some sample data into the products and price_ranges tables.

```
INSERT INTO products (product_id, product_name, price) VALUES
(1, 'Product A', 150.00),
(2, 'Product B', 300.00),
(3, 'Product C', 450.00),
(4, 'Product D', 600.00),
(5, 'Product E', 750.00);

INSERT INTO price_ranges (range_id, range_name, min_price, max_price) VALUES
(1, 'Budget', 0.00, 200.00),
(2, 'Mid-Range', 200.01, 500.00),
(3, 'Premium', 500.01, 1000.00);
```

Step 3: Perform a Non-Equi Join

Now, let's perform a non-equi join to categorize each product into a price range.

```
SELECT
    p.product_name,
    p.price,
    r.range_name
FROM
    products p
JOIN
    price_ranges r
ON
    p.price BETWEEN r.min_price AND r.max_price;
```

Explanation

- products p is the first table.
- price_ranges r is the second table.
- We perform a join with the condition p.price BETWEEN r.min_price AND r.max_price to match products to their corresponding price range.

Result

The result of the query would be:

product_name	price	range_name
Product A	150.00	Budget
Product B	300.00	Mid-Range
Product C	450.00	Mid-Range
Product D	600.00	Premium
Product E	750.00	Premium

This shows each product along with its price and the corresponding price range it falls into. This simple example demonstrates how a non-equi join works by using the BETWEEN operator to join records based on a range condition.