To demonstrate the creation of an index on a table and discuss how it improves query performance, let's consider a scenario where we have a table named "employees" with columns such as "employee_id", "first_name", "last_name", and "department_id".

We'll create an index on the "department_id" column.

Creating the Index:

```
CREATE INDEX department_index ON employees (department_id);
```

This statement creates an index named "department_index" on the "department_id" column of the "employees" table. Indexes are data structures that allow for faster retrieval of rows based on the indexed columns.

Query Performance Improvement:

After creating the index, queries that involve filtering, sorting, or joining based on the "department_id" column will likely see improved performance. When executing a query that involves the "department_id", the database engine can utilize the index to quickly locate the relevant rows, rather than performing a full table scan.

For example, a query like:

```
SELECT * FROM employees WHERE department_id = 123;
```

can now benefit from the index, resulting in faster execution.

Removing the Index:

```
DROP INDEX department_index;
```

This statement removes the index named "department_index" from the "employees" table.

Impact on Query Execution:

After dropping the index, queries that heavily rely on the "department_id" column may experience degraded performance. Without the index, the database engine will need to perform full table scans or use alternative, potentially less efficient, methods to retrieve the desired rows.

It's important to note that while indexes can improve query performance for certain types of queries, they also have associated overhead during data modification operations (such as INSERT, UPDATE, DELETE). Therefore, indexes should be carefully chosen and maintained based on the specific workload and query patterns of the database application.