Assignment 7: Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Let's assume we have a table named employees with columns employee_id, first_name, last_name, and department_id. We'll create an index on the department_id column.

```
-- Create an index on the 'department_id' column of the 'employees' table
CREATE INDEX idx_department_id ON employees(department_id);
```

Now, let's discuss how this index improves query performance:

Faster Data Retrieval: When you query data based on the department_id, the index allows the database engine to quickly locate the rows that match the specified department_id without having to scan the entire table sequentially. This results in faster data retrieval.

Reduced Disk I/O: With the index in place, the database engine can access the relevant data directly through the index structure, reducing the need for disk I/O operations. This is particularly beneficial for large tables where scanning the entire table would be resource-intensive.

Improved Query Execution Plans: The presence of an index provides the database optimizer with additional options for generating efficient query execution plans. It may choose to utilize the index to perform index scans or index seeks, resulting in optimized query performance.

Support for Join Operations: Indexes can also improve the performance of join operations, especially when joining tables on indexed columns. The database engine can use the indexes to efficiently locate matching rows, reducing the time required to process the join.

Now, let's demonstrate the impact of dropping the index on query execution:

```
-- Drop the index on the 'department_id' column
DROP INDEX idx_department_id ON employees;
```

After dropping the index, you may notice a degradation in query performance, especially for queries that heavily rely on filtering or sorting based on the department_id column. Without the index, the database engine may need to resort to full table scans or other less efficient access methods, leading to slower query execution times.

In summary, creating indexes on appropriate columns can significantly improve query performance by facilitating faster data retrieval, reducing disk I/O, enabling efficient query execution plans, and supporting join operations. However, it's essential to consider the trade-offs, such as increased storage requirements and potential overhead in data modification operations, when creating indexes.