

Task1:

The screenshot shows a database query editor interface. At the top, there is a toolbar with icons for file operations, filters, and execution. Below the toolbar, the 'Query' tab is active, displaying a single query:

```
1 EXPLAIN ANALYZE SELECT NAME FROM customer WHERE address = '2579 Joel Green Suite 253 North Russell, PA 40970';
```

Below the query editor, the 'Data Output' tab is active, showing the 'QUERY PLAN' for the executed query. The plan consists of four steps:

Step	Operation
1	Index Scan using customer_address_hash on customer (cost=0.00..8.02 rows=1 width=14) (actual time=0.030..0.030 rows=0 loops=0)
2	Index Cond: (address = '2579 Joel Green Suite 253 North Russell, PA 40970'::text)
3	Planning Time: 6.760 ms
4	Execution Time: 0.050 ms

The screenshot shows the same database query editor interface. The 'Query' tab now displays three queries:

```
1 CREATE INDEX customer_name_btree ON customer USING btree(name);
2 CREATE INDEX customer_address_hash ON customer USING hash(address);
3 EXPLAIN ANALYZE SELECT NAME FROM customer WHERE address = '2579 Joel Green Suite 253 North Russell, PA 40970';
4
```

Below the query editor, the 'Data Output' tab is active, showing the 'QUERY PLAN' for the executed query (line 3). The plan consists of four steps:

Step	Operation
1	Index Scan using customer_address_hash on customer (cost=0.00..8.02 rows=1 width=14) (actual time=0.016..0.017 rows=0 loops=0)
2	Index Cond: (address = '2579 Joel Green Suite 253 North Russell, PA 40970'::text)
3	Planning Time: 0.192 ms
4	Execution Time: 0.042 ms

adding indexes improved the performance of queries that use the indexed columns. In contrast, queries that do not use the indexed columns may not have a noticeable

improvement in performance or may even be slower due to the additional overhead of maintaining the indexes.