1. The cost of searching the names without indexing is 4390:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer.name FROM customer WHERE customer.name = 'Lindsey Webb'
2
```

Data Output

	QUERY PLAN text
1	Seq Scan on customer (cost=0.00..4390.00 rows=2 width=14) (actual time=0.009..11.812 rows=1 loops=1)
2	Filter: (name = 'Lindsey Webb'::text)
3	Rows Removed by Filter: 99999
4	Planning Time: 0.314 ms
5	Execution Time: 11.823 ms

And the cost of searching the addresses without indexing is 4390 as well:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer.address FROM customer WHERE customer.address = 'Lindsey Webb'
2
```

Data Output

	QUERY PLAN text
1	Seq Scan on customer (cost=0.00..4390.00 rows=1 width=47) (actual time=13.603..13.604 rows=0 loops=1)
2	Filter: (address = 'Lindsey Webb'::text)
3	Rows Removed by Filter: 100000
4	Planning Time: 0.139 ms
5	Execution Time: 13.615 ms

2. After applying names indexing using b-tree access method, the value of costs decreases to 8.45:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer.name FROM customer WHERE customer.name = 'Lindsey Webb'
2
```

Data Output

	QUERY PLAN text
1	Index Only Scan using name_btree on customer (cost=0.42..8.45 rows=2 width=14) (actual time=0.207..0.209 row...
2	Index Cond: (name = 'Lindsey Webb'::text)
3	Heap Fetches: 0
4	Planning Time: 3.535 ms
5	Execution Time: 0.228 ms

After doing the same to the addresses, the values of costs behaved the same way, becoming 8.44:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer.address FROM customer WHERE customer.address = 'Lindsey Webb'
2
```

Data Output

	QUERY PLAN text
1	Index Only Scan using address_btree on customer (cost=0.42..8.44 rows=1 width=47) (actual time=0.134..0.135 r...
2	Index Cond: (address = 'Lindsey Webb'::text)
3	Heap Fetches: 0
4	Planning Time: 0.119 ms
5	Execution Time: 0.154 ms

3. After applying name indexing using hash method, the value of total cost decreased as well and became 11.89:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer1.name FROM customer1 WHERE customer1.name = 'Lindsey Webb'
2
```

Data Output

	QUERY PLAN text
1	Bitmap Heap Scan on customer1 (cost=4.02..11.89 rows=2 width=14) (actual time=0.006..0.007 rows=0 loops=1)
2	Recheck Cond: (name = 'Lindsey Webb'::text)
3	-> Bitmap Index Scan on name_hash (cost=0.00..4.01 rows=2 width=0) (actual time=0.005..0.005 rows=0 loops=1)
4	Index Cond: (name = 'Lindsey Webb'::text)
5	Planning Time: 0.071 ms
6	Execution Time: 0.022 ms

After doing the same to the addresses, the value of costs decreased as well, becoming 8.02:

customers/postgres@PostgreSQL 13

Query Editor

```
1  EXPLAIN Analyze SELECT customer1.address FROM customer1 WHERE customer1.address = '06666 Campbell Wells
2  Lake Candacehaven, TX 91497-6398'
3
```

Data Output

Scratch Pa

	QUERY PLAN text
1	Index Scan using address_hash on customer1 (cost=0.00..8.02 rows=1 width=47) (actual time=0.038..0.040 rows=1 loops=1)
2	Index Cond: (address = '06666 Campbell Wells
3	Lake Candacehaven, TX 91497-6398'::text)
4	Planning Time: 0.206 ms
5	Execution Time: 0.075 ms

Conclusion: having indexes for the elements of a database increases the performance of searching. The method chosen foe searching does not really affect the performance, though, hashing addresses in my case happened to have a greater value of costs.