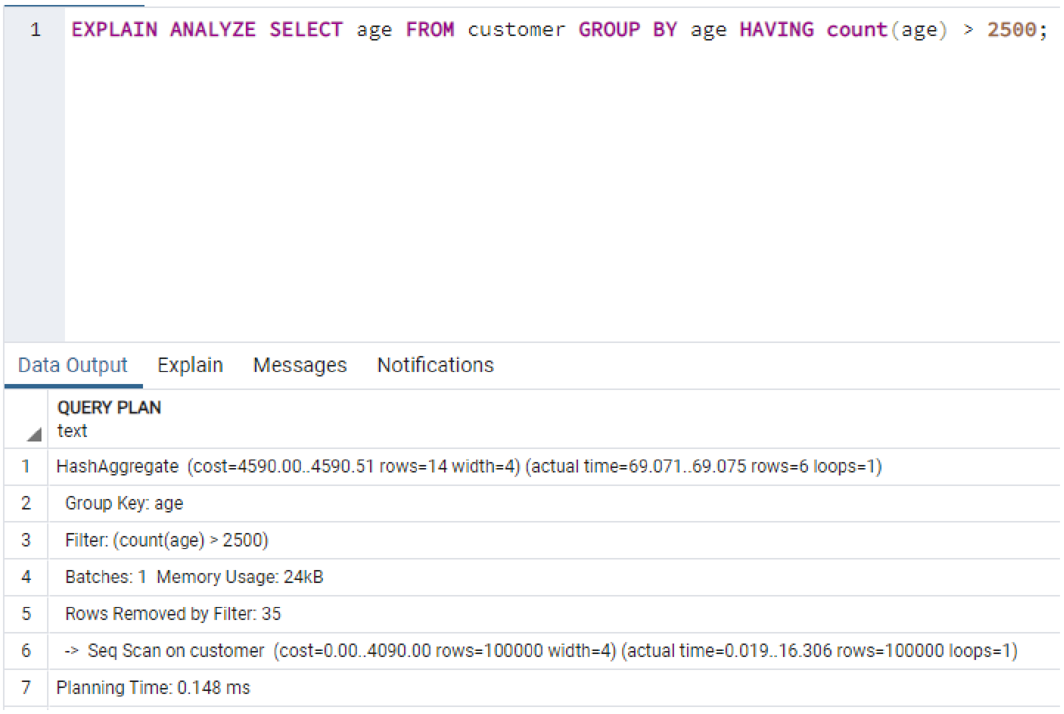
Lab 5. Anna Startseva BS19-04

I created the database "customers" using python script (warm_up.py). You can see it in repository.

The first query shows all ages that are repeated more than 2500 times. See its cost in "Seq Scan...":

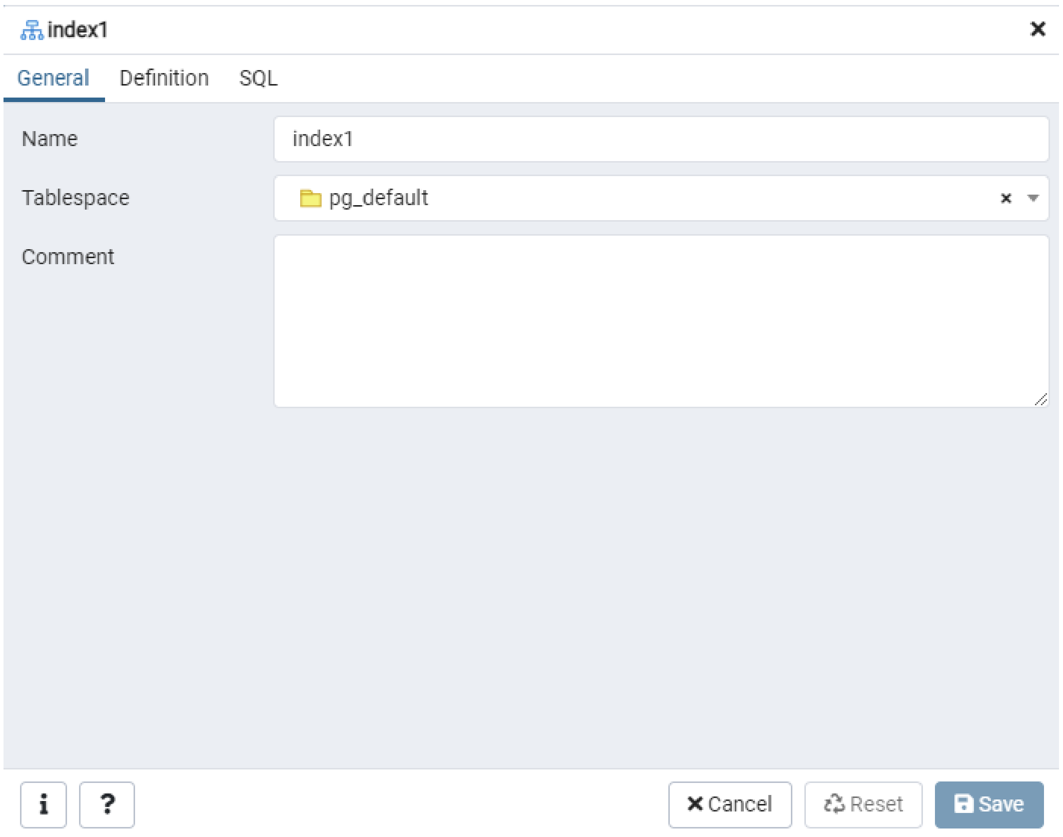


```
EXPLAIN ANALYZE SELECT age FROM customer GROUP BY age HAVING count(age) > 2500;
```

Data Output | Explain | Messages | Notifications

	QUERY PLAN text
1	HashAggregate (cost=4590.00..4590.51 rows=14 width=4) (actual time=69.071..69.075 rows=6 loops=1)
2	Group Key: age
3	Filter: (count(age) > 2500)
4	Batches: 1 Memory Usage: 24kB
5	Rows Removed by Filter: 35
6	-> Seq Scan on customer (cost=0.00..4090.00 rows=100000 width=4) (actual time=0.019..16.306 rows=100000 loops=1)
7	Planning Time: 0.148 ms

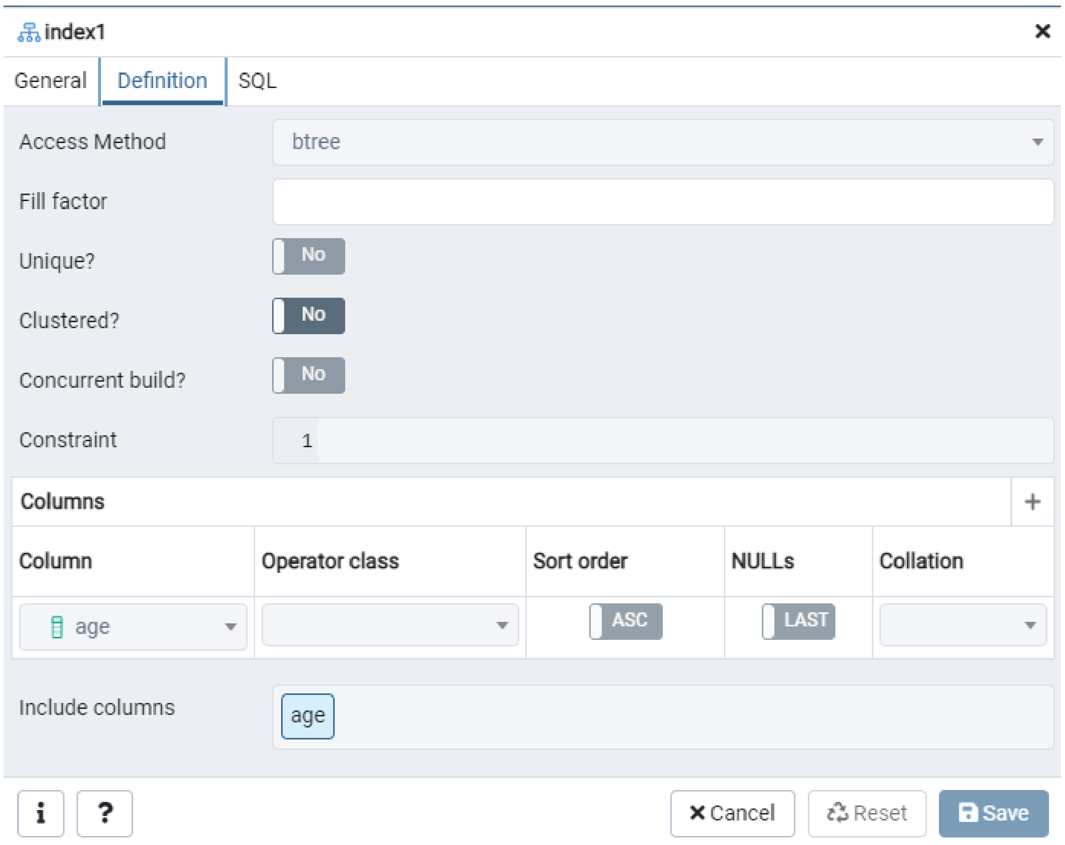
Creating a b-tree index on the column "age":



index1

General | Definition | SQL

Name	index1
Tablespace	pg_default
Comment	



index1

General | Definition | SQL

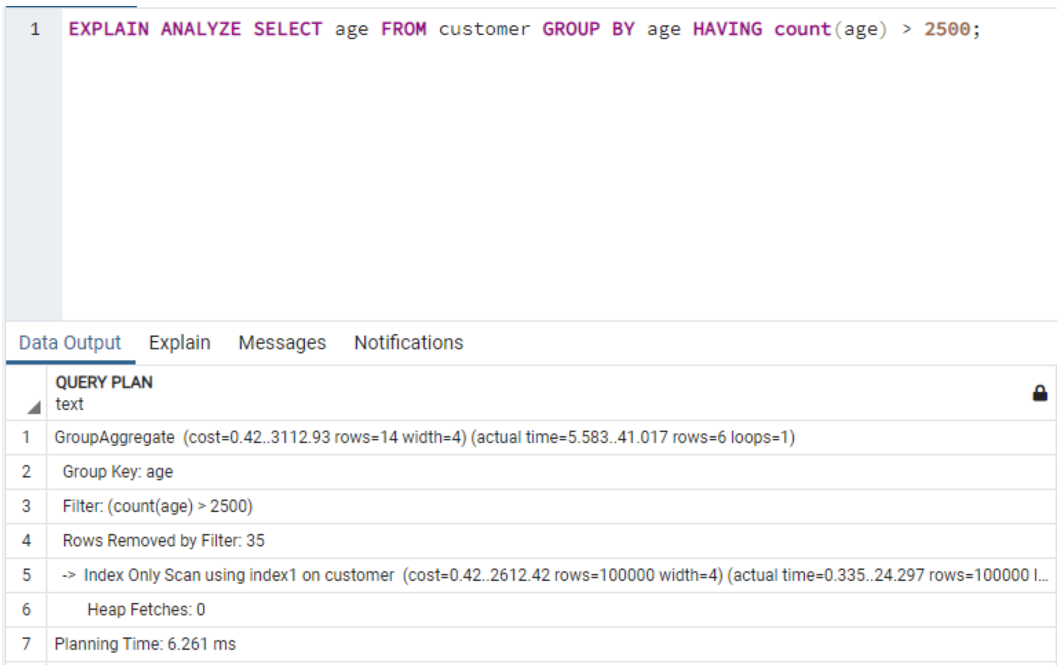
Access Method	btree
Fill factor	
Unique?	No
Clustered?	No
Concurrent build?	No
Constraint	1

Columns

Column	Operator class	Sort order	NULLs	Collation
age		ASC	LAST	

Include columns	age

The cost of the query after creating b-tree index is lower:

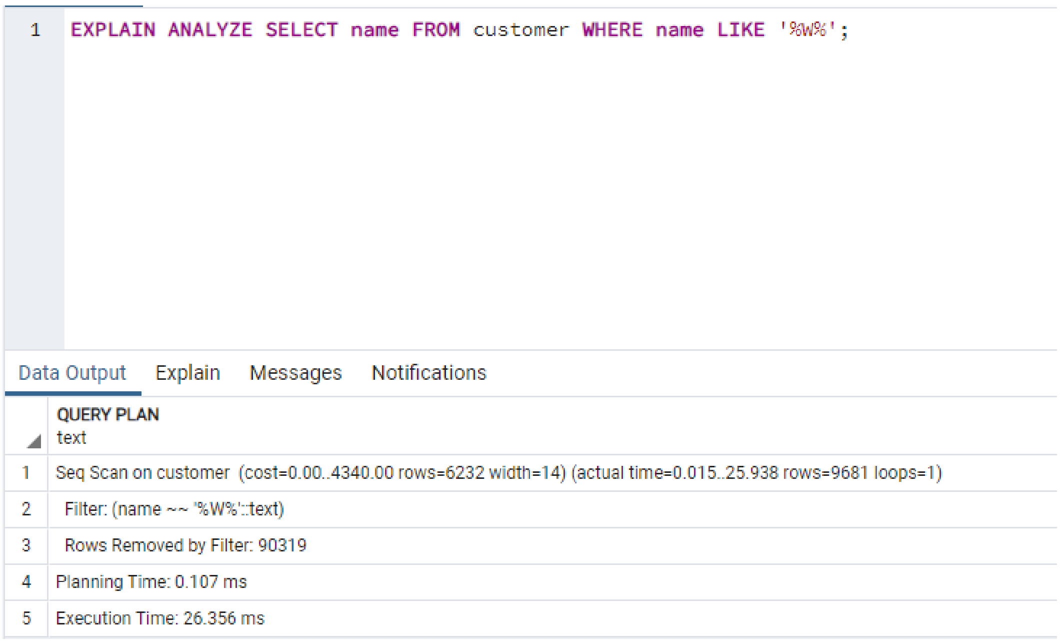


```sql
EXPLAIN ANALYZE SELECT age FROM customer GROUP BY age HAVING count(age) > 2500;
```

Data Output   Explain   Messages   Notifications

	QUERY PLAN text
1	GroupAggregate (cost=0.42..3112.93 rows=14 width=4) (actual time=5.583..41.017 rows=6 loops=1)
2	Group Key: age
3	Filter: (count(age) > 2500)
4	Rows Removed by Filter: 35
5	-> Index Only Scan using index1 on customer (cost=0.42..2612.42 rows=100000 width=4) (actual time=0.335..24.297 rows=100000 l...
6	Heap Fetches: 0
7	Planning Time: 6.261 ms

The second query shows all names of the customers which contain letter 'W'. See its cost in "Seq Scan...":

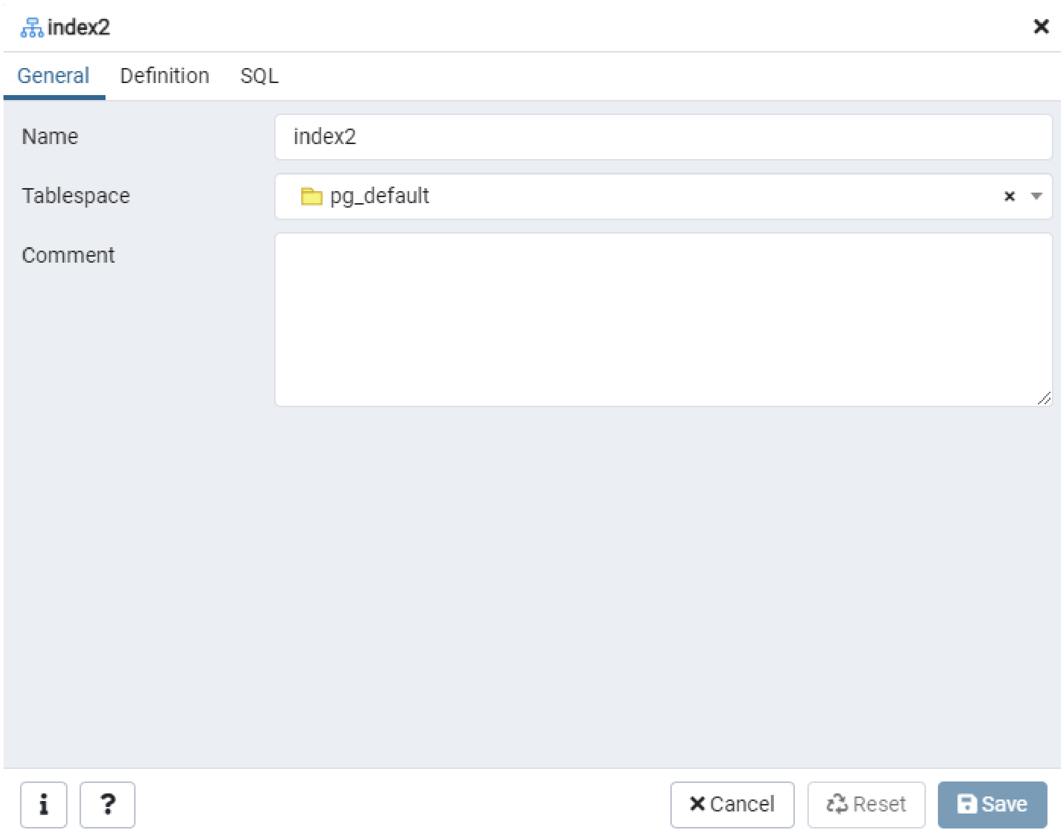


```sql
EXPLAIN ANALYZE SELECT name FROM customer WHERE name LIKE '%W%';
```

Data Output   Explain   Messages   Notifications

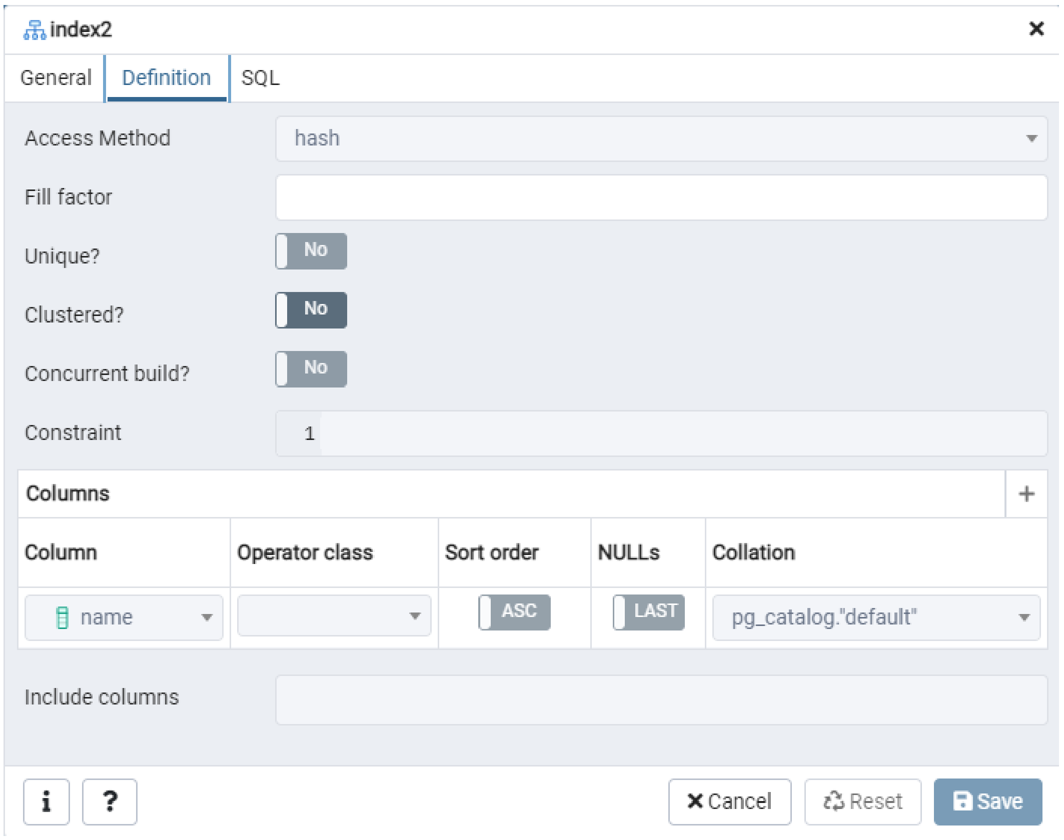
	QUERY PLAN text
1	Seq Scan on customer (cost=0.00..4340.00 rows=6232 width=14) (actual time=0.015..25.938 rows=9681 loops=1)
2	Filter: (name ~~ '%W%'::text)
3	Rows Removed by Filter: 90319
4	Planning Time: 0.107 ms
5	Execution Time: 26.356 ms

Creating a hash index on the column "name":


index2

General | Definition | SQL

Name	index2
Tablespace	pg_default
Comment	


index2

General | Definition | SQL

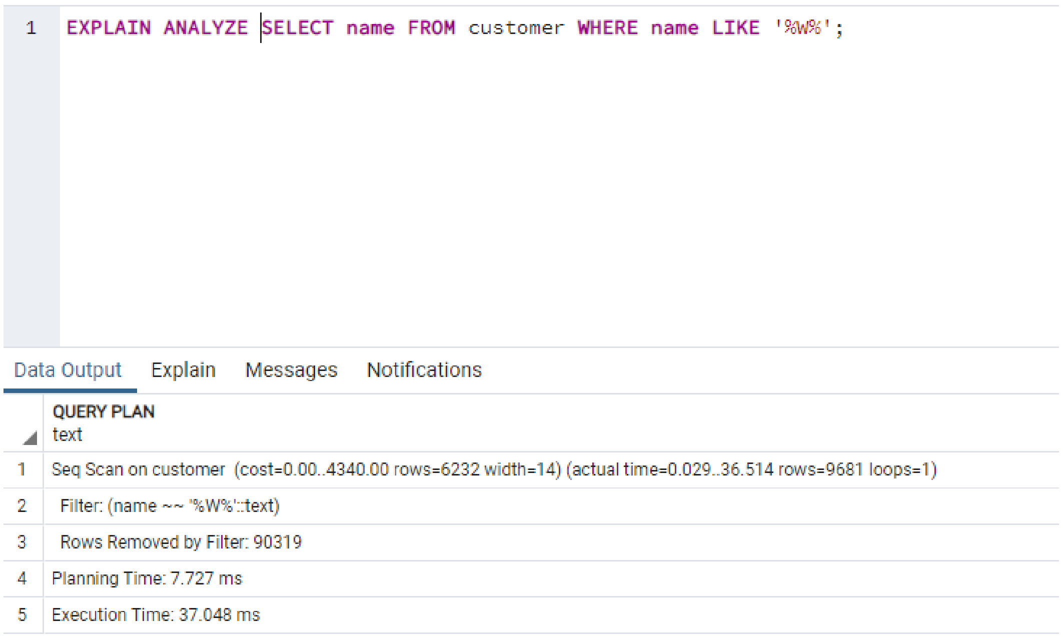
Access Method	hash
Fill factor	
Unique?	No
Clustered?	No
Concurrent build?	No
Constraint	1

Columns

Column	Operator class	Sort order	NULLs	Collation
name		ASC	LAST	pg_catalog."default"

Include columns

The cost of the query after creating index doesn't change because hash indexes only effective when there is '=' operation:

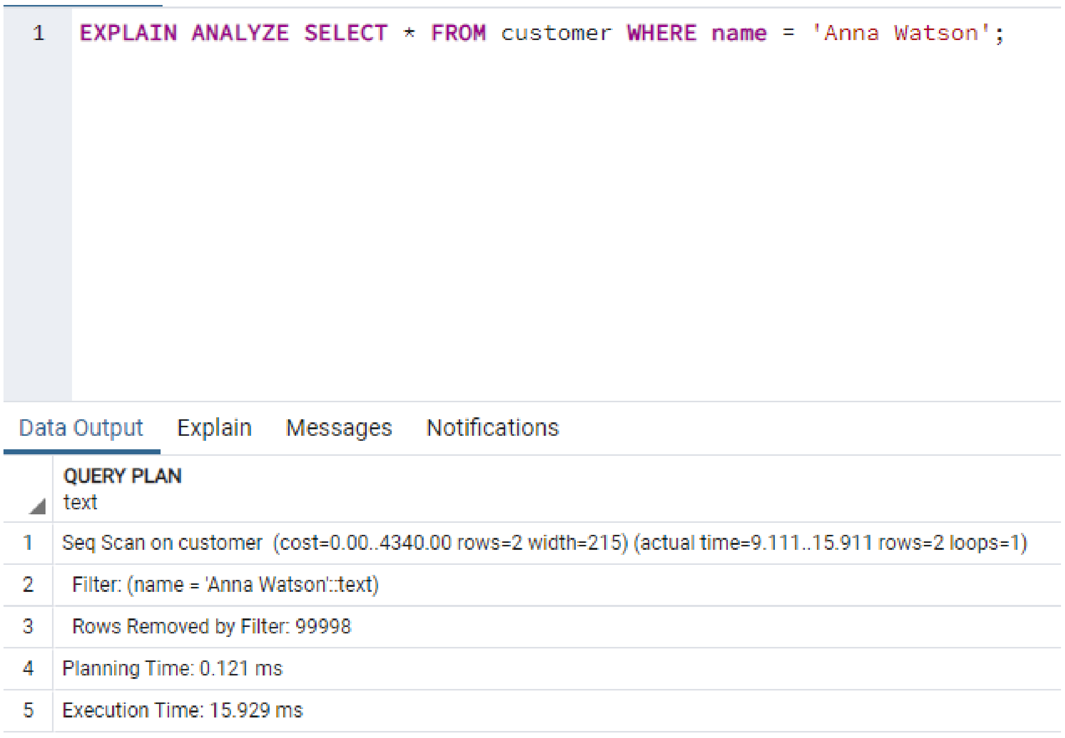


```
EXPLAIN ANALYZE SELECT name FROM customer WHERE name LIKE '%W%';
```

Data Output Explain Messages Notifications

```
QUERY PLAN
text
1  Seq Scan on customer  (cost=0.00..4340.00 rows=6232 width=14) (actual time=0.029..36.514 rows=9681 loops=1)
2    Filter: (name ~~ '%W%'::text)
3    Rows Removed by Filter: 90319
4  Planning Time: 7.727 ms
5  Execution Time: 37.048 ms
```

So, I created the query with '=' operation that finds all customers with name "Anna Watson". See its cost in "Seq Scan...":

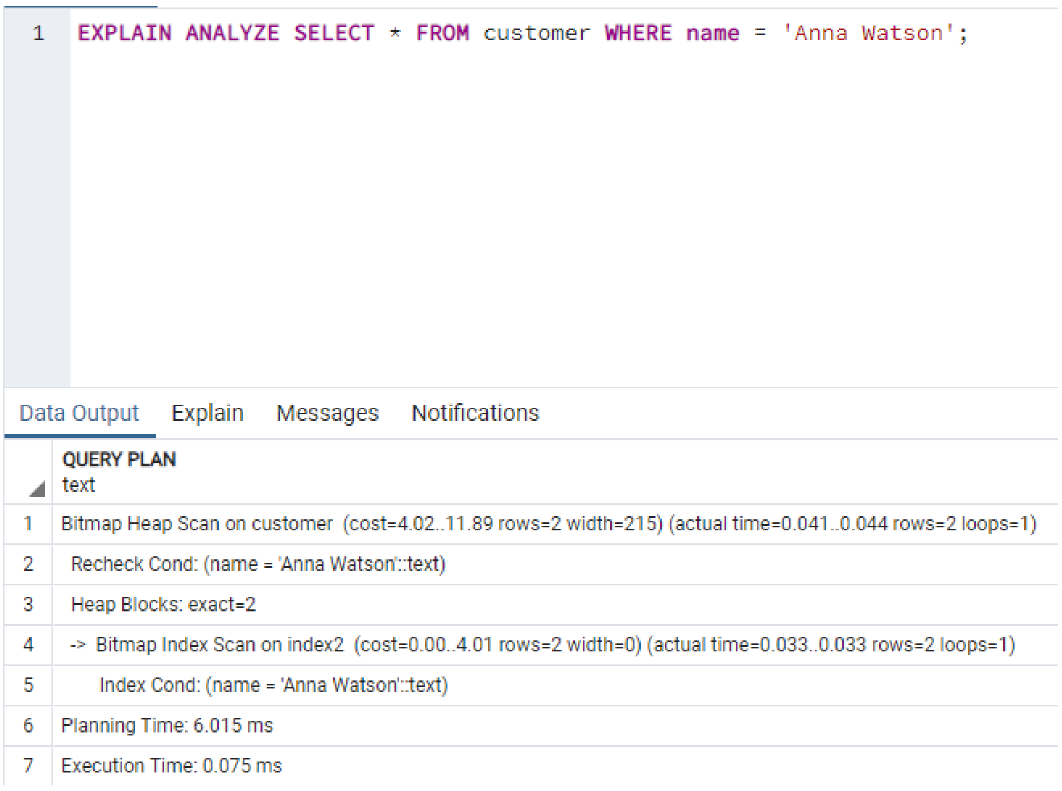


```
EXPLAIN ANALYZE SELECT * FROM customer WHERE name = 'Anna Watson';
```

Data Output Explain Messages Notifications

```
QUERY PLAN
text
1  Seq Scan on customer  (cost=0.00..4340.00 rows=2 width=215) (actual time=9.111..15.911 rows=2 loops=1)
2    Filter: (name = 'Anna Watson'::text)
3    Rows Removed by Filter: 99998
4  Planning Time: 0.121 ms
5  Execution Time: 15.929 ms
```

The cost of the query after creating hash index is lower:



```sql
EXPLAIN ANALYZE SELECT * FROM customer WHERE name = 'Anna Watson';
```

```
1  Bitmap Heap Scan on customer  (cost=4.02..11.89 rows=2 width=215) (actual time=0.041..0.044 rows=2 loops=1)
2    Recheck Cond: (name = 'Anna Watson'::text)
3    Heap Blocks: exact=2
4    ->  Bitmap Index Scan on index2  (cost=0.00..4.01 rows=2 width=0) (actual time=0.033..0.033 rows=2 loops=1)
5          Index Cond: (name = 'Anna Watson'::text)
6  Planning Time: 6.015 ms
7  Execution Time: 0.075 ms
```