Query 1: union + subquery
Default:

```
-> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.001..0.146 rows=1808 loops=1)
    -> Union materialize with deduplication  (actual time=7.757..8.094 rows=1808 loops=1)
        -> Filter: (food.Calories < (select #2))  (cost=63.64 rows=614) (actual time=0.504..1.728 rows=1842 loops=1)
            -> Table scan on food  (cost=63.64 rows=1842) (actual time=0.057..1.000 rows=1842 loops=1)
```

CREATE Index Idx on Food(Food_name):

```
-> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.001..0.147 rows=1808 loops=1)
    -> Union materialize with deduplication  (actual time=7.190..7.528 rows=1808 loops=1)
        -> Filter: (food.Calories < (select #2))  (cost=63.64 rows=614) (actual time=0.461..1.482 rows=1842 loops=1)
            -> Table scan on food  (cost=63.64 rows=1842) (actual time=0.015..0.772 rows=1842 loops=1)
```

CREATE Index Idx on Customer(name):

```
-> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.001..0.185 rows=1808 loops=1)
    -> Union materialize with deduplication  (actual time=7.487..7.865 rows=1808 loops=1)
        -> Filter: (food.Calories < (select #2))  (cost=63.64 rows=614) (actual time=0.463..1.500 rows=1842 loops=1)
            -> Table scan on food  (cost=63.64 rows=1842) (actual time=0.016..0.780 rows=1842 loops=1)
```

CREATE Index Idx on Customer(favorite_food):

```
-> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.001..0.153 rows=1808 loops=1)
    -> Union materialize with deduplication  (actual time=7.499..7.842 rows=1808 loops=1)
        -> Filter: (food.Calories < (select #2))  (cost=63.64 rows=614) (actual time=0.467..1.567 rows=1842 loops=1)
            -> Table scan on food  (cost=63.64 rows=1842) (actual time=0.016..0.820 rows=1842 loops=1)
```

Analysis:
Based on our data, we see that index based on Food Name is slightly faster.
We think that its because food_name has many similar food names, so in terms of hashing,
There is less bucket splitting in the table, thus it is slightly faster to lookup the data.