

Indexing:

Query 4:

ORDER BY AvgCaloriesPerNutritionalType DESC;

```
EXPLAIN ANALYZE
```

```
SELECT f.NutritionType,
```

```
•   AVG(f.CaloriesPerGram * f.Quantity) AS  
AvgCaloriesPerNutritionalType
```

```
FROM Food f
```

```
GROUP BY f.NutritionType
```

```
HAVING AVG(f.CaloriesPerGram * f.Quantity) > (
```

```
    SELECT AVG(f2.CaloriesPerGram * f2.Quantity)
```

```
    FROM Food f2
```

```
)
```

```
UNION
```

```
SELECT d.NutritionType,
```

```
•   AVG(d.CaloriesPerGram * d.Quantity) AS  
AvgCaloriesPerNutritionalType
```

```
FROM Drink d
```

```

GROUP BY d.NutritionType

HAVING AVG(d.CaloriesPerGram * d.Quantity) > (

SELECT AVG(d2.CaloriesPerGram * d2.Quantity)

FROM Drink d2

)

```

The result shows that:

Index 1: Adding Index on NutritionType in Food and Drink

```

-> Sort: AvgCaloriesPerNutritionalType DESC (cost=2.60..2.60 rows=0) (actual time=2.687..2.688 rows=4 loops=1)
-> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=2.678..2.679 rows=4 loops=1)
-> Union materialize with deduplication (cost=0.00..0.00 rows=0) (actual time=2.678..2.678 rows=4 loops=1)
-> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (actual time=1.519..1.520 rows=1 loops=1)

```

Since NutritionType is used in GROUP BY and affects aggregation performance, an index on NutritionType may improve query performance.

```
CREATE INDEX idx_food_nutrition_type ON Food (NutritionType);
```

```
CREATE INDEX idx_drink_nutrition_type ON Drink (NutritionType);
```

result:

```

-> Sort: AvgCaloriesPerNutritionalType DESC (cost=2880.84..2880.84 rows=1913) (actual time=3.846..3.847 rows=4 loops=1)
-> Table scan on <union temporary> (cost=577.66..604.06 rows=1913) (actual time=3.835..3.836 rows=4 loops=1)
-> Union materialize with deduplication (cost=577.65..577.65 rows=1913) (actual time=3.834..3.834 rows=4 loops=1)
-> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (cost=184.35 rows=913) (actual time=1.799..1.970 rows=1 loops=1)

```

Index 2: Adding Index on CaloriesPerGram in Food and Drink

Since CaloriesPerGram is used in calculations within the HAVING clause, an index on this attribute might reduce the cost of filtering.

```
CREATE INDEX idx_food_calories_per_gram ON Food (CaloriesPerGram);
```

```
CREATE INDEX idx_drink_calories_per_gram ON Drink (CaloriesPerGram);
```

Result:

```
-> Sort: AvgCaloriesPerNutritionalType DESC (cost=2880.84..2880.84 rows=1913) (actual time=3.765..3.765 rows=4 loops=1)
-> Table scan on <union temporary> (cost=577.66..604.06 rows=1913) (actual time=3.753..3.753 rows=4 loops=1)
-> Union materialize with deduplication (cost=577.65..577.65 rows=1913) (actual time=3.751..3.751 rows=4 loops=1)
-> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (cost=184.35 rows=913) (actual time=1.723..1.878)
```

index 3: Combined Index on NutritionType and CaloriesPerGram in Food and Drink

composite indexes on both NutritionType and CaloriesPerGram if these attributes appear frequently in queries.

```
CREATE INDEX idx_food_nutrition_calories ON Food (NutritionType, CaloriesPerGram);
```

```
CREATE INDEX idx_drink_nutrition_calories ON Drink (NutritionType, CaloriesPerGram);
```

result:

```
EXPLAIN: -> Sort: AvgCaloriesPerNutritionalType DESC (cost=2880.84..2880.84 rows=1913) (actual time=4.142..4.142 rows=4 loops=1)
-> Table scan on <union temporary> (cost=577.66..604.06 rows=1913) (actual time=4.127..4.128 rows=4 loops=1)
-> Union materialize with deduplication (cost=577.65..577.65 rows=1913) (actual time=4.119..4.119 rows=4 loops=1)
-> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (cost=184.35 rows=913) (actual time=1.920..2.093)
```

From all these 3 results, we discover that these indexes are not effective on this query on time.

Query 3:

```
SELECT f.NutritionType,
```

- `AVG(f.CaloriesPerGram * f.Quantity) AS AvgCaloriesPerNutritionalType`

```
FROM Food f
```



```
GROUP BY f.NutritionType
```



```
HAVING AVG(f.CaloriesPerGram * f.Quantity) > (
```



```
    SELECT AVG(f2.CaloriesPerGram * f2.Quantity)
```



```
    FROM Food f2
```



```
)
```

UNION

```
SELECT d.NutritionType,
```


- `AVG(d.CaloriesPerGram * d.Quantity) AS AvgCaloriesPerNutritionalType`

```
FROM Drink d
```



```
GROUP BY d.NutritionType
```



```
HAVING AVG(d.CaloriesPerGram * d.Quantity) > (
```



```
    SELECT AVG(d2.CaloriesPerGram * d2.Quantity)
```



```
    FROM Drink d2
```

)

```
ORDER BY AvgCaloriesPerNutritionalType DESC;
```

Baseline:

```
EXPLAIN:      -> Sort: AvgCaloriesPerNutritionalType DESC (cost=2.60..2.60 rows=0) (actual time=2.505..2.505 rows=4 loops=1)
               -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=2.496..2.496 rows=4 loops=1)
                   -> Union materialize with deduplication (cost=0.00..0.00 rows=0) (actual time=2.495..2.495 rows=4 loops=1)
                       -> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (actual time=1.373..1.374 rows=1 loops=1)
```

Index 1: Single Index on NutritionType in Food

```
CREATE INDEX idx_food_nutrition_type ON Food (NutritionType);
```

```
EXPLAIN:      -> Sort: AvgCaloriesPerNutritionalType DESC (cost=1278.75..1278.75 rows=913) (actual time=3.234..3.234 rows=4 loops=1)
               -> Table scan on <union temporary> (cost=275.67..289.56 rows=913) (actual time=3.224..3.225 rows=4 loops=1)
                   -> Union materialize with deduplication (cost=275.65..275.65 rows=913) (actual time=3.224..3.224 rows=4 loops=1)
                       -> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (cost=184.35 rows=913) (actual time=1.899..2.119)
```

We saw that it has no effect.

```
CREATE INDEX idx_food_calories_per_gram ON Food (CaloriesPerGram);
```

```
CREATE INDEX idx_drink_calories_per_gram ON Drink (CaloriesPerGram);
```

```
CREATE INDEX idx_food_calories_per_gram ON Food (CaloriesPerGram);
```

```
CREATE INDEX idx_drink_calories_per_gram ON Drink (CaloriesPerGram);
```

Index 2: Single Index on CaloriesPerGram in Food and Drink

```
EXPLAIN:      -> Sort: AvgCaloriesPerNutritionalType DESC (cost=2.60..2.60 rows=0) (actual time=2.502..2.502 rows=4 loops=1)
               -> Table scan on <union temporary> (cost=2.50..2.50 rows=0) (actual time=2.491..2.492 rows=4 loops=1)
                   -> Union materialize with deduplication (cost=0.00..0.00 rows=0) (actual time=2.491..2.491 rows=4 loops=1)
                       -> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (actual time=1.356..1.356 rows=1 loops=1)
```

We saw that it has slightly effect.

Index 3: Combined Index on NutritionType and CaloriesPerGram in Food and Drink

```
CREATE INDEX idx_food_nutrition_calories ON Food (NutritionType,  
CaloriesPerGram);  
CREATE INDEX idx_drink_nutrition_calories ON Drink (NutritionType,  
CaloriesPerGram);
```

EXPLAIN:

```
-> Sort: AvgCaloriesPerNutritionalType DESC (cost=2880.84..2880.84 rows=1913) (actual time=3.655..3.655 rows=4 loops=1)  
-> Table scan on <union temporary> (cost=577.66..604.06 rows=1913) (actual time=3.645..3.645 rows=4 loops=1)  
-> Union materialize with deduplication (cost=577.65..577.65 rows=1913) (actual time=3.643..3.643 rows=4 loops=1)  
-> Filter: (avg((f.CaloriesPerGram * f.Quantity)) > (select #2)) (cost=184.35 rows=913) (actual time=1.639..1.803)
```

We saw that it has no effect.