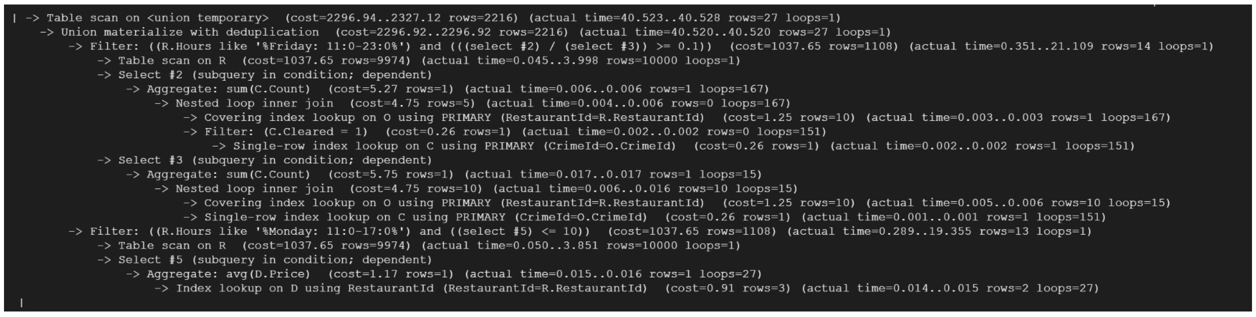
Indexing on both Restaurant(Hours) and Crime(Cleared): Cost 2296.94..2327.12

Create:

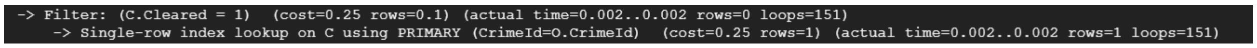
CREATE INDEX Hours_idx ON Restaurant(Hours);

CREATE INDEX Cleared_idx ON Crime(Cleared);



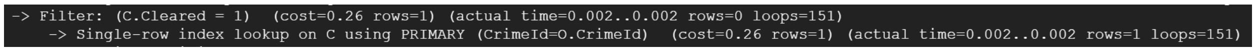
```
| -> Table scan on <union temporary>  (cost=2296.94..2327.12 rows=2216) (actual time=40.523..40.528 rows=27 loops=1)
    -> Union materialize with deduplication  (cost=2296.92..2296.92 rows=2216) (actual time=40.520..40.520 rows=27 loops=1)
        -> Filter: ((R.Hours like '%Friday: 11:0-23:0%') and (((select #2) / (select #3)) >= 0.1))  (cost=1037.65 rows=1108) (actual time=0.351..21.109 rows=14 loops=1)
            -> Table scan on R  (cost=1037.65 rows=9974) (actual time=0.045..3.998 rows=10000 loops=1)
            -> Select #2 (subquery in condition; dependent)
                -> Aggregate: sum(C.Count)  (cost=5.27 rows=1) (actual time=0.006..0.006 rows=1 loops=167)
                    -> Nested loop inner join  (cost=4.75 rows=5) (actual time=0.004..0.006 rows=0 loops=167)
                        -> Covering index lookup on O using PRIMARY (RestaurantId=R.RestaurantId)  (cost=1.25 rows=10) (actual time=0.003..0.003 rows=1 loops=167)
                        -> Filter: (C.Cleared = 1)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=0 loops=151)
                            -> Single-row index lookup on C using PRIMARY (CrimeId=O.CrimeId)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=151)
            -> Select #3 (subquery in condition; dependent)
                -> Aggregate: sum(C.Count)  (cost=5.75 rows=1) (actual time=0.017..0.017 rows=1 loops=15)
                    -> Nested loop inner join  (cost=4.75 rows=10) (actual time=0.006..0.016 rows=10 loops=15)
                        -> Covering index lookup on O using PRIMARY (RestaurantId=R.RestaurantId)  (cost=1.25 rows=10) (actual time=0.005..0.006 rows=10 loops=15)
                        -> Single-row index lookup on C using PRIMARY (CrimeId=O.CrimeId)  (cost=0.26 rows=1) (actual time=0.001..0.001 rows=1 loops=151)
        -> Filter: ((R.Hours like '%Monday: 11:0-17:0%') and ((select #5) <= 10))  (cost=1037.65 rows=1108) (actual time=0.289..19.355 rows=13 loops=1)
            -> Table scan on R  (cost=1037.65 rows=9974) (actual time=0.050..3.851 rows=10000 loops=1)
            -> Select #5 (subquery in condition; dependent)
                -> Aggregate: avg(D.Price)  (cost=1.17 rows=1) (actual time=0.015..0.016 rows=1 loops=27)
                    -> Index lookup on D using RestaurantId (RestaurantId=R.RestaurantId)  (cost=0.91 rows=3) (actual time=0.014..0.015 rows=2 loops=27)
 |
```

where



```
-> Filter: (C.Cleared = 1)  (cost=0.25 rows=0.1) (actual time=0.002..0.002 rows=0 loops=151)
    -> Single-row index lookup on C using PRIMARY (CrimeId=O.CrimeId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=151)
```

is replaced by



```
-> Filter: (C.Cleared = 1)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=0 loops=151)
    -> Single-row index lookup on C using PRIMARY (CrimeId=O.CrimeId)  (cost=0.26 rows=1) (actual time=0.002..0.002 rows=1 loops=151)
```

The result is expected from the result of both indexing above. The total cost does not change at all as adding each of the indexes separately does. The increase in time from 0.25 to 0.26 after adding the two indexes can be similarly explained as above. It is worth noting that there is no other indexing schema that can have a good chance of increasing the query performance. This is because the others are either in select statement or in order by, which does not help to improve the overall performance.

Drop:

DROP INDEX Hours_idx ON Restaurant;

DROP INDEX Cleared_idx ON Crime;