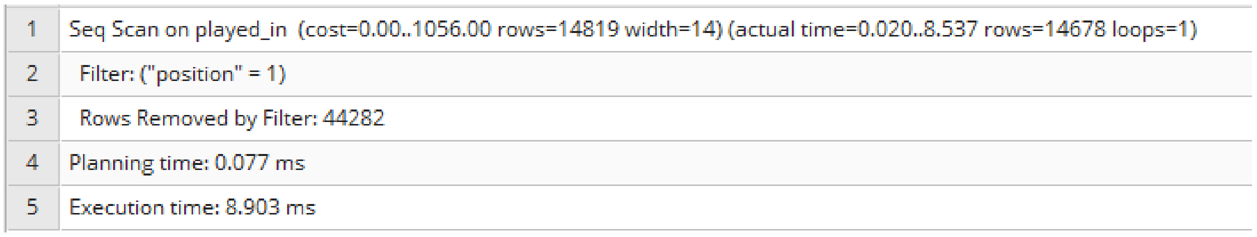
Part A

Q1.

*EXPLAIN ANALYZE SELECT * from played_in WHERE position=1;*

Execution Plan: Sequential Scan



```
Seq Scan on played_in  (cost=0.00..1056.00 rows=14819 width=14) (actual time=0.020..8.537 rows=14678 loops=1)
  Filter: ("position" = 1)
  Rows Removed by Filter: 44282
Planning time: 0.077 ms
Execution time: 8.903 ms
```

Q2.

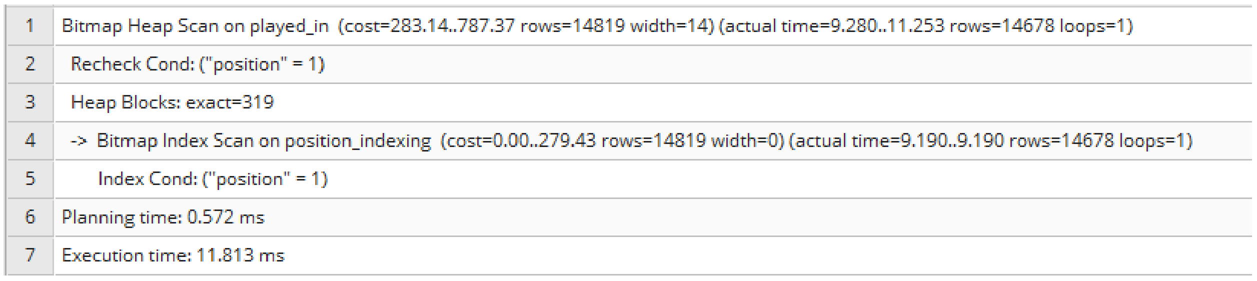
Estimated Cost for this query with the above execution plan was 0.00 + 1056.00.

Q3.

CREATE INDEX position_indexing *ON* played_in (position);

*EXPLAIN ANALYZE SELECT * from played_in WHERE position=1;*

Execution Plan: Bitmap Heap Scan



```
Bitmap Heap Scan on played_in  (cost=283.14..787.37 rows=14819 width=14) (actual time=9.280..11.253 rows=14678 loops=1)
  Recheck Cond: ("position" = 1)
  Heap Blocks: exact=319
  -> Bitmap Index Scan on position_indexing  (cost=0.00..279.43 rows=14819 width=0) (actual time=9.190..9.190 rows=14678 loops=1)
       Index Cond: ("position" = 1)
Planning time: 0.572 ms
Execution time: 11.813 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 283.14 + 787.37.

Q5.

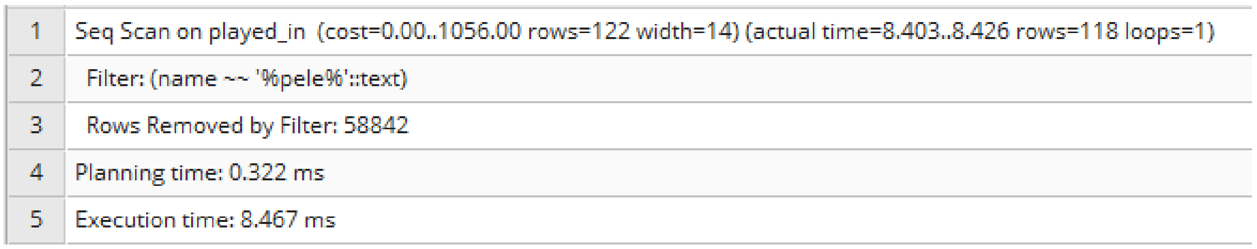
Yes, the cost changed from 0.00 + 1056.00 to 283.14 + 787.37 because the bitmap index scan was used. Instead of brute-force iterations over the data.

Part B

Q1.

*EXPLAIN ANALYZE SELECT * from played_in WHERE name like '%pele%';*

Execution Plan: Sequential Scan



```
Seq Scan on played_in  (cost=0.00..1056.00 rows=122 width=14) (actual time=8.403..8.426 rows=118 loops=1)
  Filter: (name ~~ '%pele%'::text)
  Rows Removed by Filter: 58842
Planning time: 0.322 ms
Execution time: 8.467 ms
```

Q2.

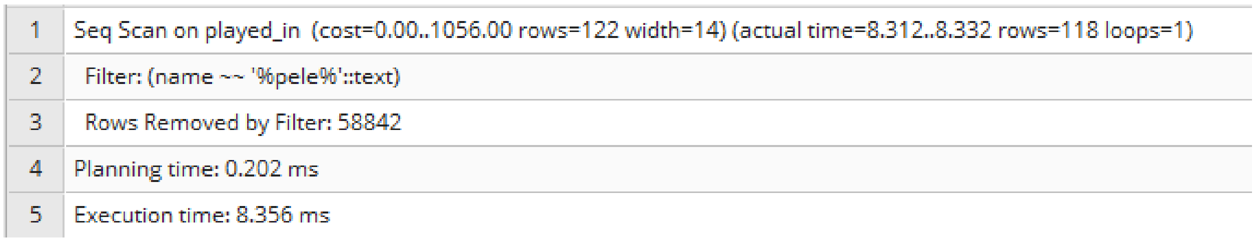
Estimated Cost for this query with the above execution plan was 0.00 + 1056.00.

Q3.

CREATE INDEX name_indexing *ON* played_in (name);

*EXPLAIN ANALYZE SELECT * from played_in WHERE name like '%pele%';*

Execution Plan: Sequential Scan



```
Seq Scan on played_in  (cost=0.00..1056.00 rows=122 width=14) (actual time=8.312..8.332 rows=118 loops=1)
  Filter: (name ~~ '%pele%'::text)
  Rows Removed by Filter: 58842
Planning time: 0.202 ms
Execution time: 8.356 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 0.00 + 1056.00.

Q5.

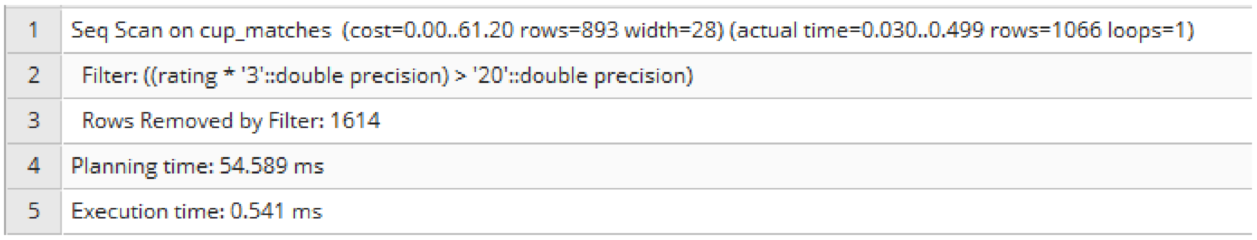
No, the cost did not change as no indexing was needed for this query because unification over strings was performed with all possible prefixes and suffixes, therefore no indexing was needed and brute-force iteration over the data is used.

Part C

Q1.

*EXPLAIN ANALYZE SELECT * from cup_matches WHERE rating*3 > 20;*

Execution Plan: Sequential Scan



```
Seq Scan on cup_matches  (cost=0.00..61.20 rows=893 width=28) (actual time=0.030..0.499 rows=1066 loops=1)
  Filter: ((rating * '3'::double precision) > '20'::double precision)
  Rows Removed by Filter: 1614
Planning time: 54.589 ms
Execution time: 0.541 ms
```

Q2.

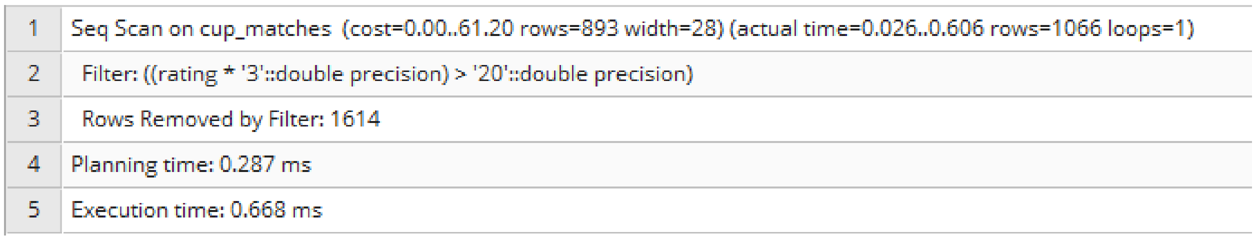
Estimated Cost for this query with the above execution plan was 0.00 + 61.20.

Q3.

CREATE INDEX rating_indexing ON cup_matches (rating);

*EXPLAIN ANALYZE SELECT * from cup_matches WHERE rating*3 > 20;*

Execution Plan: Sequential Scan



```
Seq Scan on cup_matches  (cost=0.00..61.20 rows=893 width=28) (actual time=0.026..0.606 rows=1066 loops=1)
  Filter: ((rating * '3'::double precision) > '20'::double precision)
  Rows Removed by Filter: 1614
Planning time: 0.287 ms
Execution time: 0.668 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 0.00 + 61.20.

Q5.

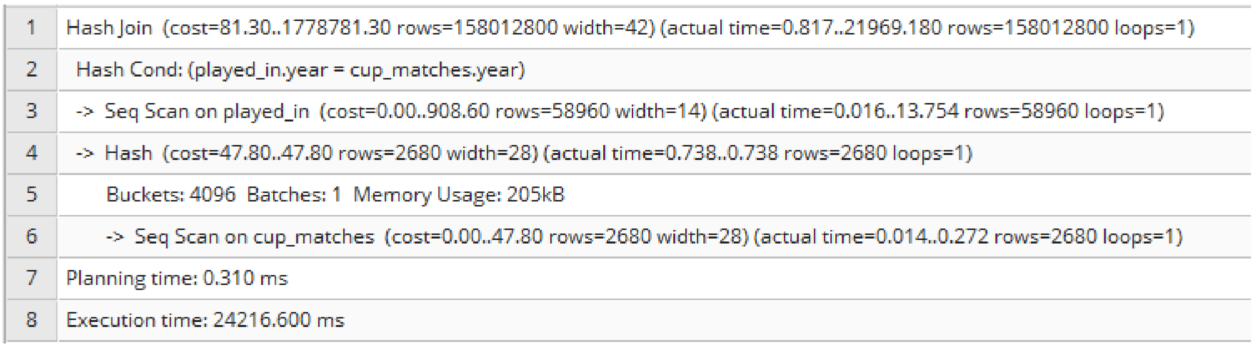
No, the cost did not change as no indexing was needed for this query because index is only available for direct comparisons, not post-evaluation comparisons.

Part D

Q1.

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.year=played_in.year;*

Execution Plan: Sequential Scan



```
Hash Join  (cost=81.30..1778781.30 rows=158012800 width=42) (actual time=0.817..21969.180 rows=158012800 loops=1)
 Hash Cond: (played_in.year = cup_matches.year)
 ->  Seq Scan on played_in  (cost=0.00..908.60 rows=58960 width=14) (actual time=0.016..13.754 rows=58960 loops=1)
 ->  Hash  (cost=47.80..47.80 rows=2680 width=28) (actual time=0.738..0.738 rows=2680 loops=1)
       Buckets: 4096  Batches: 1  Memory Usage: 205kB
       ->  Seq Scan on cup_matches  (cost=0.00..47.80 rows=2680 width=28) (actual time=0.014..0.272 rows=2680 loops=1)
Planning time: 0.310 ms
Execution time: 24216.600 ms
```

Q2.

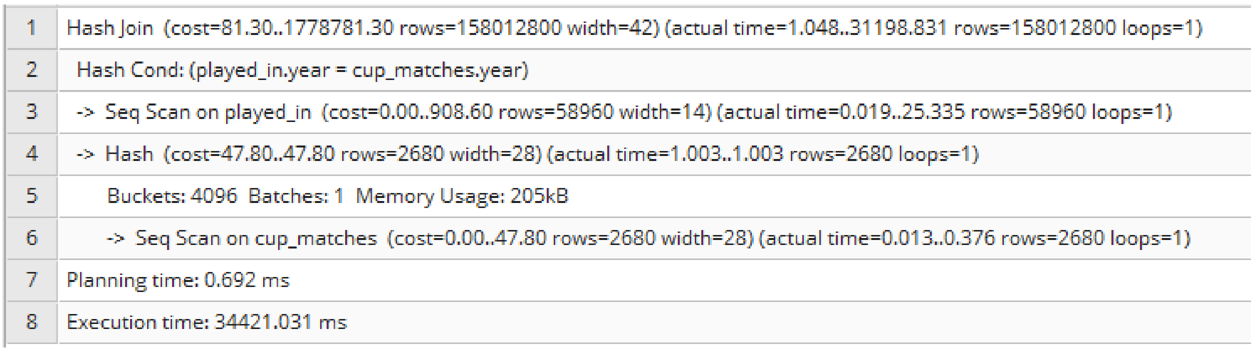
Estimated Cost for this query with the above execution plan was 81.30 + 1778781.30.

Q3.

CREATE INDEX year_indexing ON cup_matches (year);

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.year=played_in.year;*

Execution Plan: Sequential Scan



```
Hash Join  (cost=81.30..1778781.30 rows=158012800 width=42) (actual time=1.048..31198.831 rows=158012800 loops=1)
 Hash Cond: (played_in.year = cup_matches.year)
 ->  Seq Scan on played_in  (cost=0.00..908.60 rows=58960 width=14) (actual time=0.019..25.335 rows=58960 loops=1)
 ->  Hash  (cost=47.80..47.80 rows=2680 width=28) (actual time=1.003..1.003 rows=2680 loops=1)
       Buckets: 4096  Batches: 1  Memory Usage: 205kB
       ->  Seq Scan on cup_matches  (cost=0.00..47.80 rows=2680 width=28) (actual time=0.013..0.376 rows=2680 loops=1)
Planning time: 0.692 ms
Execution time: 34421.031 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 81.30 + 1778781.30.

Q5.

No, the cost did not change as no indexing was needed for this query because of the equality condition for over the indexed parameter (yea), thus a Sequential Scan must be performed with no use of the created index.

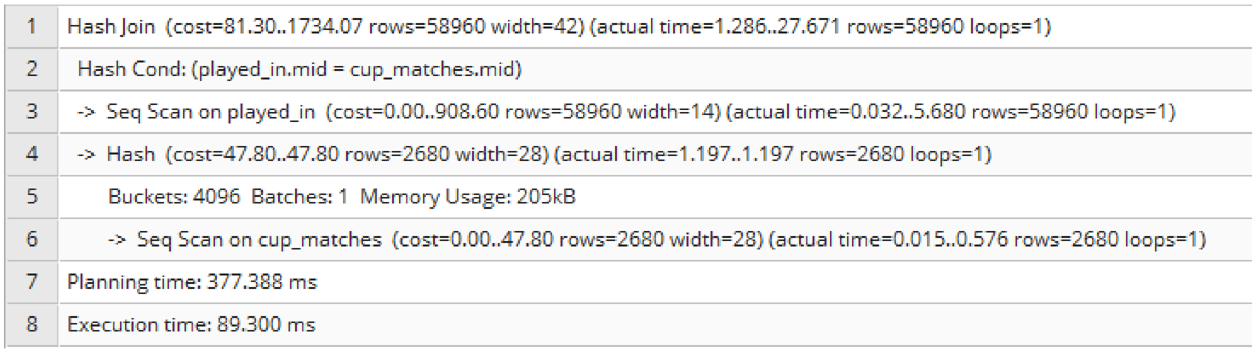
Part E

Q1.

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Sequential Scan

Join Algorithm: Hash Join



```
Hash Join  (cost=81.30..1734.07 rows=58960 width=42) (actual time=1.286..27.671 rows=58960 loops=1)
  Hash Cond: (played_in.mid = cup_matches.mid)
  ->  Seq Scan on played_in  (cost=0.00..908.60 rows=58960 width=14) (actual time=0.032..5.680 rows=58960 loops=1)
  ->  Hash  (cost=47.80..47.80 rows=2680 width=28) (actual time=1.197..1.197 rows=2680 loops=1)
        Buckets: 4096  Batches: 1  Memory Usage: 205kB
        ->  Seq Scan on cup_matches  (cost=0.00..47.80 rows=2680 width=28) (actual time=0.015..0.576 rows=2680 loops=1)
Planning time: 377.388 ms
Execution time: 89.300 ms
```

Estimated Cost for this query with the above execution plan was 81.30 + 1734.07.

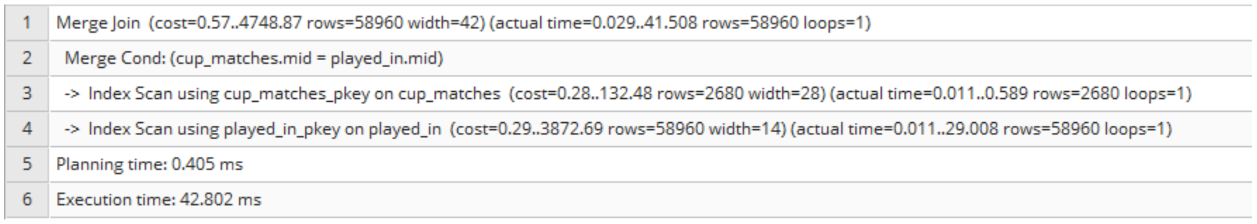
Q2.

SET enable_hashjoin=false;

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Index Scan

Join Algorithm: Merge Join



```
Merge Join  (cost=0.57..4748.87 rows=58960 width=42) (actual time=0.029..41.508 rows=58960 loops=1)
  Merge Cond: (cup_matches.mid = played_in.mid)
  ->  Index Scan using cup_matches_pkey on cup_matches  (cost=0.28..132.48 rows=2680 width=28) (actual time=0.011..0.589 rows=2680 loops=1)
  ->  Index Scan using played_in_pkey on played_in  (cost=0.29..3872.69 rows=58960 width=14) (actual time=0.011..29.008 rows=58960 loops=1)
Planning time: 0.405 ms
Execution time: 42.802 ms
```

Estimated Cost for this query with the above execution plan was 0.57 + 4748.87.

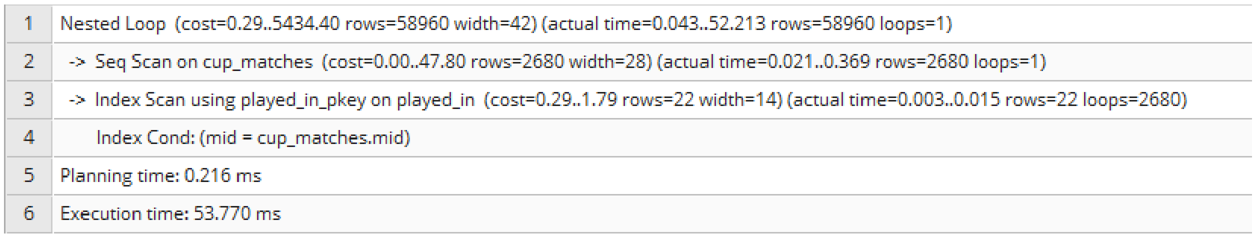
Q3.

SET enable_mergejoin=false;

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Sequential Scan

Join Algorithm: Nested Loop



```
Nested Loop  (cost=0.29..5434.40 rows=58960 width=42) (actual time=0.043..52.213 rows=58960 loops=1)
  -> Seq Scan on cup_matches  (cost=0.00..47.80 rows=2680 width=28) (actual time=0.021..0.369 rows=2680 loops=1)
  -> Index Scan using played_in_pkey on played_in  (cost=0.29..1.79 rows=22 width=14) (actual time=0.003..0.015 rows=22 loops=2680)
       Index Cond: (mid = cup_matches.mid)
Planning time: 0.216 ms
Execution time: 53.770 ms
```

Estimated Cost for this query with the above execution plan was 81.30 + 1778781.30.

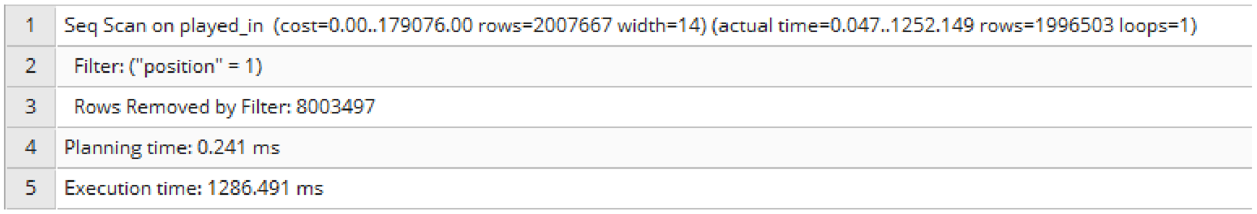
Part F

Part A

Q1.

*EXPLAIN ANALYZE SELECT * from played_in WHERE position=1;*

Execution Plan: Gather - Parallel Sequential Scan



```
Seq Scan on played_in  (cost=0.00..179076.00 rows=2007667 width=14) (actual time=0.047..1252.149 rows=1996503 loops=1)
  Filter: ("position" = 1)
  Rows Removed by Filter: 8003497
Planning time: 0.241 ms
Execution time: 1286.491 ms
```

Q2.

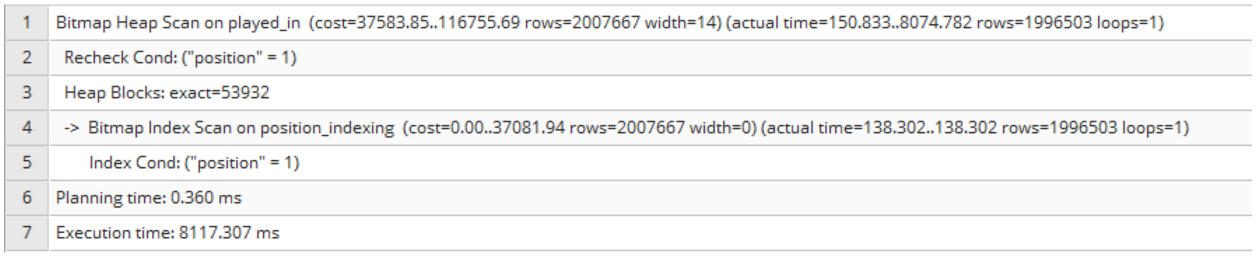
Estimated Cost for this query with the above execution plan was 0.00 + 179076.00.

Q3.

CREATE INDEX position_indexing *ON* played_in (position);

*EXPLAIN ANALYZE SELECT * from played_in WHERE position=1;*

Execution Plan: Bitmap Heap Scan



```
Bitmap Heap Scan on played_in  (cost=37583.85..116755.69 rows=2007667 width=14) (actual time=150.833..8074.782 rows=1996503 loops=1)
  Recheck Cond: ("position" = 1)
  Heap Blocks: exact=53932
  ->  Bitmap Index Scan on position_indexing  (cost=0.00..37081.94 rows=2007667 width=0) (actual time=138.302..138.302 rows=1996503 loops=1)
        Index Cond: ("position" = 1)
Planning time: 0.360 ms
Execution time: 8117.307 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 37583.85 + 116755.69.

Q5.

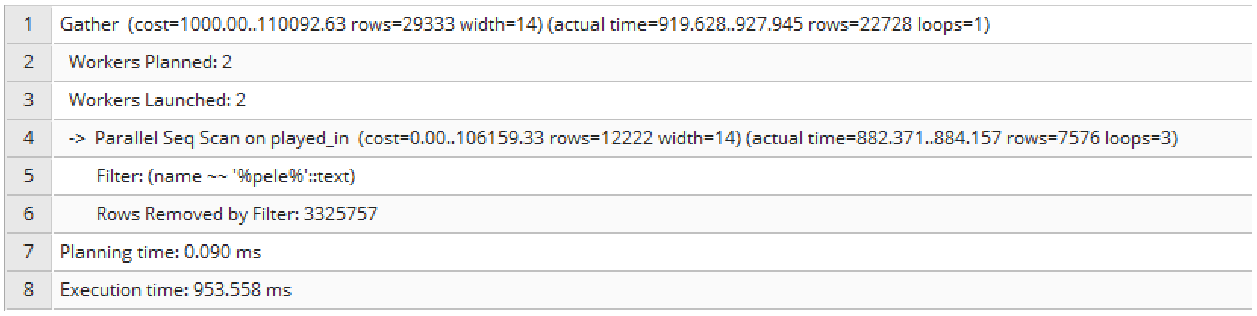
Yes, the cost changed from 0.00 + 179076.00 to 37583.85 + 116755.69 because the bitmap index scan was used. Instead of brute-force iterations over the data.

Part B

Q1.

*EXPLAIN ANALYZE SELECT * from played_in WHERE name like '%pele%';*

Execution Plan: Gather - Parallel Sequential Scan



```
Gather  (cost=1000.00..110092.63 rows=29333 width=14) (actual time=919.628..927.945 rows=22728 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Parallel Seq Scan on played_in  (cost=0.00..106159.33 rows=12222 width=14) (actual time=882.371..884.157 rows=7576 loops=3)
        Filter: (name ~~ '%pele%'::text)
        Rows Removed by Filter: 3325757
Planning time: 0.090 ms
Execution time: 953.558 ms
```

Q2.

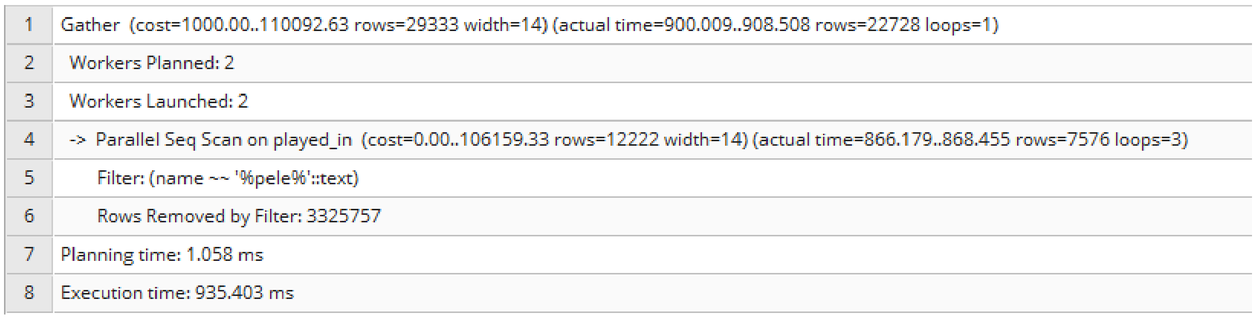
Estimated Cost for this query with the above execution plan was 1000.00 + 110092.63.

Q3.

CREATE INDEX name_indexing *ON* played_in (name);

*EXPLAIN ANALYZE SELECT * from played_in WHERE name like '%pele%';*

Execution Plan: Gather – Parallel Sequential Scan



```
Gather  (cost=1000.00..110092.63 rows=29333 width=14) (actual time=900.009..908.508 rows=22728 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Parallel Seq Scan on played_in  (cost=0.00..106159.33 rows=12222 width=14) (actual time=866.179..868.455 rows=7576 loops=3)
        Filter: (name ~~ '%pele%'::text)
        Rows Removed by Filter: 3325757
Planning time: 1.058 ms
Execution time: 935.403 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 1000.00 + 110092.63.

Q5.

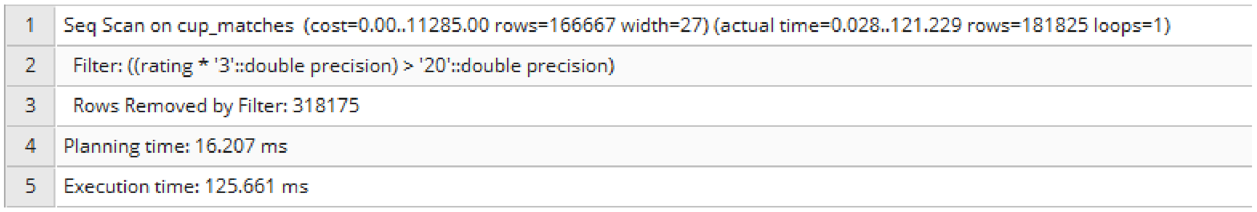
No, the cost did not change as no indexing was needed for this query because unification over strings was performed with all possible prefixes and suffixes, therefore no indexing was needed and brute-force iteration over the data is used.

Part C

Q1.

*EXPLAIN ANALYZE SELECT * from cup_matches WHERE rating*3 > 20;*

Execution Plan: Sequential Scan



```
Seq Scan on cup_matches  (cost=0.00..11285.00 rows=166667 width=27) (actual time=0.028..121.229 rows=181825 loops=1)
  Filter: ((rating * '3'::double precision) > '20'::double precision)
  Rows Removed by Filter: 318175
Planning time: 16.207 ms
Execution time: 125.661 ms
```

Q2.

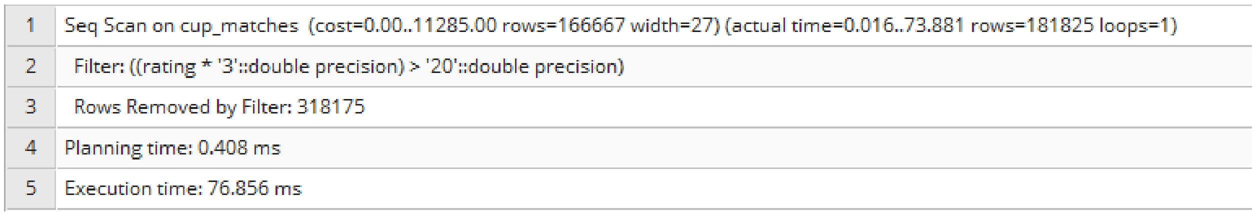
Estimated Cost for this query with the above execution plan was 0.00 + 11285.00.

Q3.

CREATE INDEX rating_indexing ON cup_matches (rating);

*EXPLAIN ANALYZE SELECT * from cup_matches WHERE rating*3 > 20;*

Execution Plan: Sequential Scan



```
Seq Scan on cup_matches  (cost=0.00..11285.00 rows=166667 width=27) (actual time=0.016..73.881 rows=181825 loops=1)
  Filter: ((rating * '3'::double precision) > '20'::double precision)
  Rows Removed by Filter: 318175
Planning time: 0.408 ms
Execution time: 76.856 ms
```

Q4.

Estimated Cost for this query with the above execution plan was 0.00 + 11285.00.

Q5.

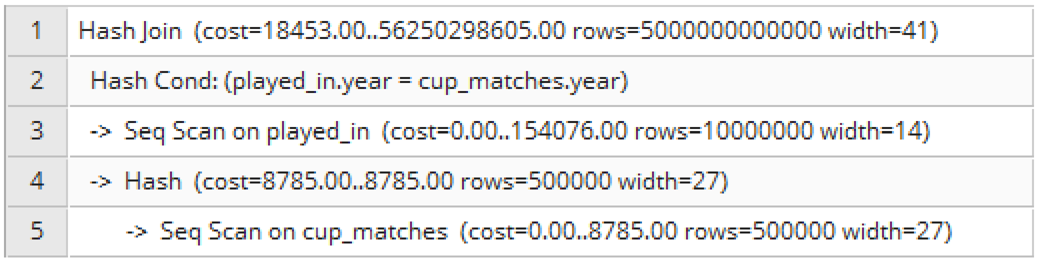
No, the cost did not change as no indexing was needed for this query because index is only available for direct comparisons, not post-evaluation comparisons.

Part D

Q1.

*EXPLAIN SELECT * from cup_matches, played_in WHERE cup_matches.year=played_in.year;*

Execution Plan: Sequential Scan



```
Hash Join  (cost=18453.00..56250298605.00 rows=500000000000 width=41)
  Hash Cond: (played_in.year = cup_matches.year)
  ->  Seq Scan on played_in  (cost=0.00..154076.00 rows=10000000 width=14)
  ->  Hash  (cost=8785.00..8785.00 rows=500000 width=27)
        ->  Seq Scan on cup_matches  (cost=0.00..8785.00 rows=500000 width=27)
```

Q2.

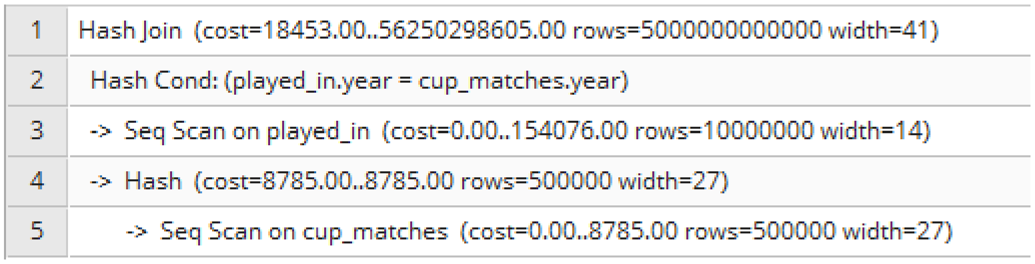
Estimated Cost for this query with the above execution plan was 18453.00 + 56250298605.00.

Q3.

CREATE INDEX year_indexing ON cup_matches (year);

*EXPLAIN ANALYZE SELECT * from cup_matches, played_in WHERE cup_matches.year=played_in.year;*

Execution Plan: Sequential Scan



```
Hash Join  (cost=18453.00..56250298605.00 rows=500000000000 width=41)
  Hash Cond: (played_in.year = cup_matches.year)
  ->  Seq Scan on played_in  (cost=0.00..154076.00 rows=10000000 width=14)
  ->  Hash  (cost=8785.00..8785.00 rows=500000 width=27)
        ->  Seq Scan on cup_matches  (cost=0.00..8785.00 rows=500000 width=27)
```

Q4.

Estimated Cost for this query with the above execution plan was 18453.00 + 56250298605.00.

Q5.

No, the cost did not change as no indexing was needed for this query because of the equality condition for over the indexed parameter (yea), thus a Sequential Scan must be performed with no use of the created index.

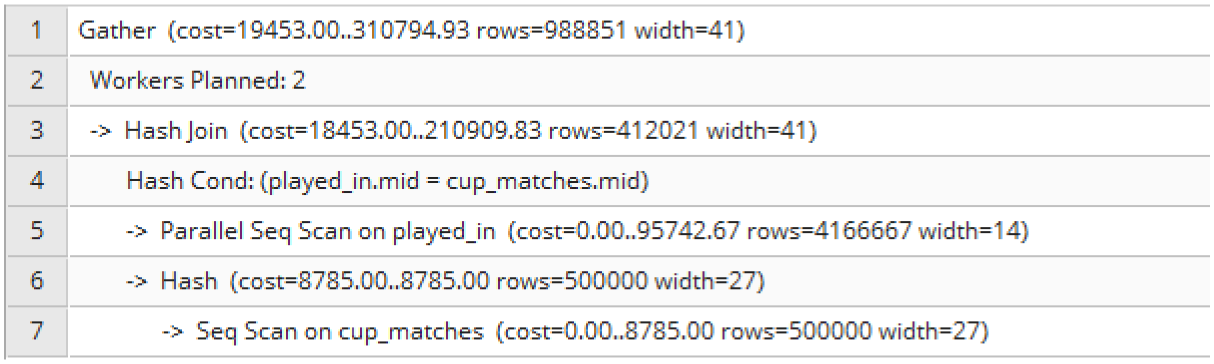
Part E

Q1.

*EXPLAIN SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Parallel Sequential Scan

Join Algorithm: Gather - Hash Join



```
Gather  (cost=19453.00..310794.93 rows=988851 width=41)
  Workers Planned: 2
  ->  Hash Join  (cost=18453.00..210909.83 rows=412021 width=41)
        Hash Cond: (played_in.mid = cup_matches.mid)
        ->  Parallel Seq Scan on played_in  (cost=0.00..95742.67 rows=4166667 width=14)
        ->  Hash  (cost=8785.00..8785.00 rows=500000 width=27)
              ->  Seq Scan on cup_matches  (cost=0.00..8785.00 rows=500000 width=27)
```

Estimated Cost for this query with the above execution plan was 19453.00 + 310794.93.

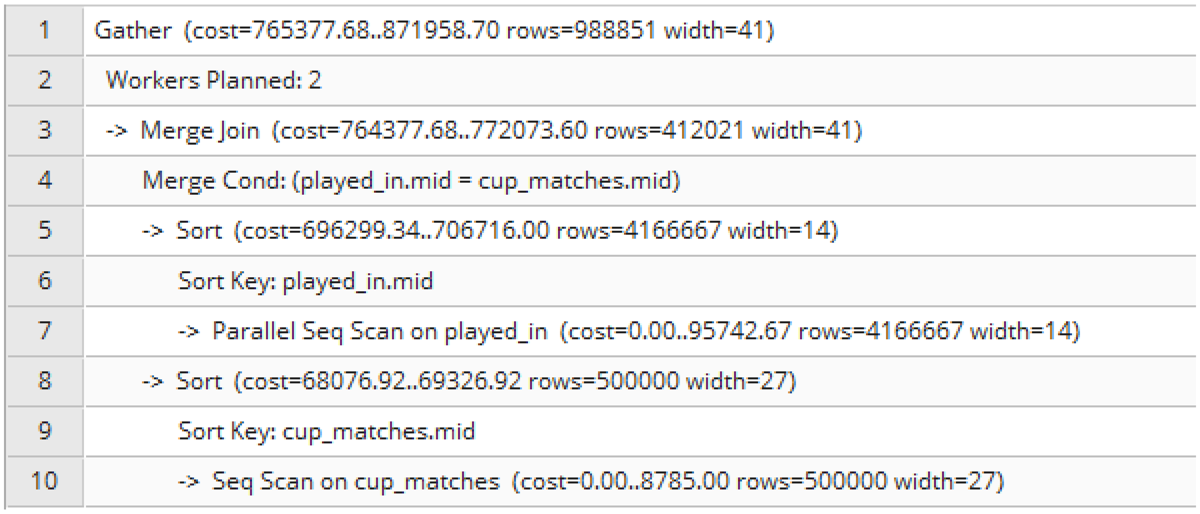
Q2.

SET enable_hashjoin=false;

*EXPLAIN SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Parallel Sequential Scan

Join Algorithm: Gather - Merge Join



```
Gather  (cost=765377.68..871958.70 rows=988851 width=41)
  Workers Planned: 2
  ->  Merge Join  (cost=764377.68..772073.60 rows=412021 width=41)
        Merge Cond: (played_in.mid = cup_matches.mid)
        ->  Sort  (cost=696299.34..706716.00 rows=4166667 width=14)
              Sort Key: played_in.mid
              ->  Parallel Seq Scan on played_in  (cost=0.00..95742.67 rows=4166667 width=14)
        ->  Sort  (cost=68076.92..69326.92 rows=500000 width=27)
              Sort Key: cup_matches.mid
              ->  Seq Scan on cup_matches  (cost=0.00..8785.00 rows=500000 width=27)
```

Estimated Cost for this query with the above execution plan was 765377.68 + 871958.70.

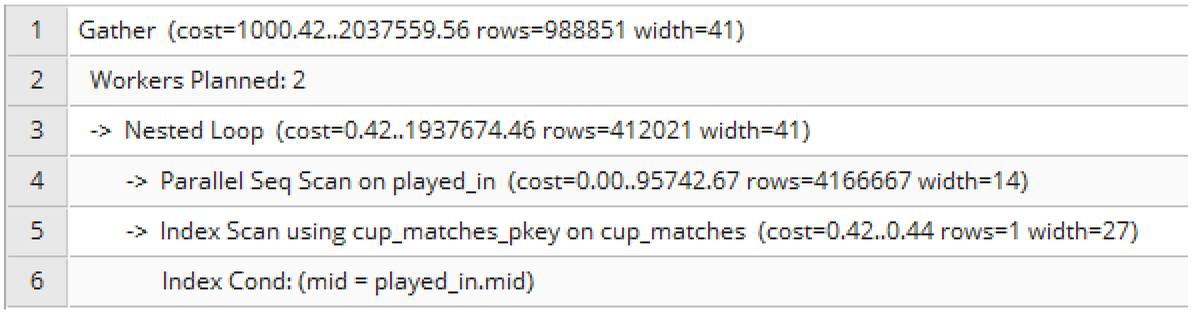
Q3.

SET enable_mergejoin=false;

*EXPLAIN SELECT * from cup_matches, played_in WHERE cup_matches.mid=played_in.mid;*

Execution Plan: Parallel Sequential Scan – Index Scan

Join Algorithm: Gather - Nested Loop



```
Gather  (cost=1000.42..2037559.56 rows=988851 width=41)
  Workers Planned: 2
  ->  Nested Loop  (cost=0.42..1937674.46 rows=412021 width=41)
        ->  Parallel Seq Scan on played_in  (cost=0.00..95742.67 rows=4166667 width=14)
        ->  Index Scan using cup_matches_pkey on cup_matches  (cost=0.42..0.44 rows=1 width=27)
              Index Cond: (mid = played_in.mid)
```

Estimated Cost for this query with the above execution plan was 1000.42 + 2037559.56.