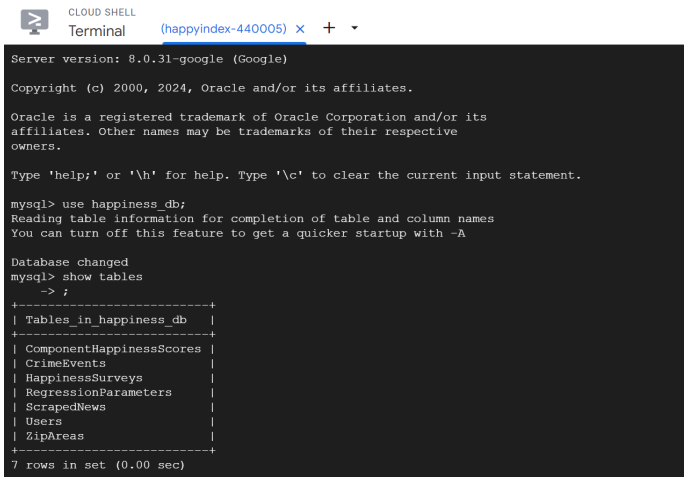
Database Implementation and Indexing

Implementation of Database in GCP



Data Definition Language (DDL) commands

Provided at the end of the **Database Design.pdf** file in /doc directory.

Insert Data into Tables

32,743 rows in ZipAreas table:

```
mysql> select count(*) from ZipAreas;
+----------+
| count(*) |
+----------+
|    32743 |
+----------+
1 row in set (0.24 sec)
```

24,062 rows in ComponentHappinessScores table:

```
mysql> select count(*) from ComponentHappinessScores;
+----------+
| count(*) |
+----------+
|    24062 |
+----------+
1 row in set (0.01 sec)
```

57,832 rows in CrimeEvents table:

```
mysql> select count(*) from CrimeEvents;
+----------+
| count(*) |
+----------+
|    57832 |
+----------+
1 row in set (0.02 sec)
```

1,000 rows (autogenerated) in Users table:

```
mysql> select count(*) from Users;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

1,000 rows (autogenerated) in HappinessSurveys table:

```
mysql> select count(*) from HappinessSurveys;
+----------+
| count(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.01 sec)
```

316 rows in ScrapedNews table:

```
mysql> select count(*) from ScrapedNews;
+----------+
| count(*) |
+----------+
|      316 |
+----------+
1 row in set (0.00 sec)
```

Advanced SQL Queries

Complex query #1

```
-- Best zipCodes of each city in a given state:
-- All zipCodes in the state of 'IL' where totalHappinessScore is the best of each
city

SELECT za.zipCode, za.city, chs.totalHappinessScore
FROM ZipAreas za
JOIN ComponentHappinessScores chs ON za.zipCode = chs.zipCode
JOIN (
    SELECT za.city, MAX(chs.totalHappinessScore) AS maxHappiness
    FROM ZipAreas za
    JOIN ComponentHappinessScores chs ON za.zipCode = chs.zipCode
    WHERE za.stateCode = 'IL'
    GROUP BY za.city
) maxScores ON za.city = maxScores.city AND chs.totalHappinessScore =
maxScores.maxHappiness
WHERE za.stateCode = 'IL';
```

```
mysql> SELECT za.zipCode, za.city, chs.totalHappinessScore FROM ZipAreas za JOIN ComponentHappinessScores chs ON za.zipCode = chs.zipCode JOIN (     SE
LECT za.city, MAX(chs.totalHappinessScore) AS maxHappiness     FROM ZipAreas za     JOIN ComponentHappinessScores chs ON za.zipCode = chs.zipCode     W
HERE za.stateCode = 'IL'     GROUP BY za.city ) maxScores ON za.city = maxScores.city AND chs.totalHappinessScore = maxScores.maxHappiness WHERE za.sta
teCode = 'IL' LIMIT 15;
+---------+------------------+---------------------+
| zipCode | city             | totalHappinessScore |
+---------+------------------+---------------------+
| 60002   | Antioch          |               63.55 |
| 60005   | Arlington Heights |               62.94 |
| 60007   | Elk Grove Village |               63.11 |
| 60008   | Rolling Meadows  |               62.52 |
| 60010   | Barrington       |               64.89 |
| 60012   | Crystal Lake     |               66.10 |
| 60013   | Cary             |               64.09 |
| 60015   | Deerfield        |               66.32 |
| 60018   | Des Plaines      |               61.37 |
| 60020   | Fox Lake         |               61.81 |
| 60021   | Fox River Grove  |               65.41 |
| 60022   | Glencoe          |               65.85 |
| 60026   | Glenview         |               65.17 |
| 60030   | Grayslake        |               62.96 |
| 60031   | Gurnee           |               63.03 |
+---------+------------------+---------------------+
15 rows in set (0.07 sec)
```

Complex query #2

```
-- Safest and Happiest user reported zip codes in Chicago:
-- All zipCodes where Users report on average above 8 happiness and where there exists no crime events from the past 30 days.

SELECT hs.zipCode
FROM HappinessSurveys hs
JOIN (
    SELECT za.zipCode, AVG(hs2.userReportedHappinessScore) AS avgHappinessScore
    FROM HappinessSurveys hs2 JOIN ZipAreas za ON za.zipCode = hs2.zipCode
    WHERE 1=1
        AND za.stateCode = 'IL'
    GROUP BY za.zipCode
    HAVING AVG(hs2.userReportedHappinessScore) > 8
) avgHappiness ON hs.zipCode = avgHappiness.zipCode
EXCEPT (
    SELECT DISTINCT ce.zipCode
    FROM CrimeEvents ce
    WHERE ce.Date ⩾ CURDATE() - INTERVAL 30 DAY
);
```

```
mysql> SELECT hs.zipCode
    -> FROM HappinessSurveys hs
    -> JOIN (
    ->     SELECT za.zipCode, AVG(hs2.userReportedHappinessScore) AS avgHappinessScore
    ->     FROM HappinessSurveys hs2 JOIN ZipAreas za ON za.zipCode = hs2.zipCode
    ->     WHERE 1=1
    ->         AND za.stateCode = 'IL'
    ->     GROUP BY za.zipCode
    ->     HAVING AVG(hs2.userReportedHappinessScore) > 8
    -> ) avgHappiness ON hs.zipCode = avgHappiness.zipCode
    -> EXCEPT (
    ->     SELECT DISTINCT ce.zipCode
    ->     FROM CrimeEvents ce
    ->     WHERE ce.Date ⩾ CURDATE() - INTERVAL 30 DAY
    -> );
+---------+
| zipCode |
+---------+
| 62861   |
| 60043   |
| 61751   |
| 61764   |
| 61951   |
| 62803   |
+---------+
```

Note! Output is less than 15 rows.

Complex query #3

```
-- Places to avoid:
-- All zipCodes where there is on average more 'ASSAULT' CrimeEvents and totalHappinessScore is below the city average in ‘Chicago, IL’

WITH AssaultCrimePerCapita AS (
    SELECT ce.zipCode, COUNT(*) / za.population AS avgAssaultPerCapita
    FROM CrimeEvents ce
    JOIN ZipAreas za ON za.zipCode = ce.zipCode
    WHERE 1=1
        AND ce.primaryType = 'ASSAULT'
        AND za.stateCode = 'IL'
        AND za.city = 'Chicago'
    GROUP BY ce.zipCode
),
ZipCodeHappiness AS (
    SELECT chs.zipCode, chs.totalHappinessScore
    FROM ComponentHappinessScores chs
    JOIN ZipAreas za ON chs.zipCode = za.zipCode
    WHERE 1=1
        AND za.stateCode = 'IL'
        AND za.city = 'Chicago'
)
SELECT acp.zipCode, acp.avgAssaultPerCapita, zch.totalHappinessScore
FROM AssaultCrimePerCapita acp
JOIN ZipCodeHappiness zch ON acp.zipCode = zch.zipCode
WHERE 1=1
    AND acp.avgAssaultPerCapita > (
        SELECT AVG(avgAssaultPerCapita)
        FROM AssaultCrimePerCapita
    )
    AND zch.totalHappinessScore < (
        SELECT AVG(totalHappinessScore)
        FROM ZipCodeHappiness
    );
```

```
mysql> WITH AssaultCrimePerCapita AS (     SELECT ce.zipCode, COUNT(*) / za.population AS avgAssaultPerCapita     FROM CrimeEvents ce     JOIN ZipAreas
 za ON za.zipCode = ce.zipCode     WHERE 1=1         AND ce.primaryType = 'ASSAULT'         AND za.stateCode = 'IL'         AND za.city = 'Chicago'
 GROUP BY ce.zipCode ), ZipCodeHappiness AS (     SELECT chs.zipCode, chs.totalHappinessScore     FROM ComponentHappinessScores chs     JOIN ZipAreas z
a ON chs.zipCode = za.zipCode     WHERE 1=1         AND za.stateCode = 'IL'         AND za.city = 'Chicago' ) SELECT acp.zipCode, acp.avgAssaultPerCapi
ta, zch.totalHappinessScore FROM AssaultCrimePerCapita acp JOIN ZipCodeHappiness zch ON acp.zipCode = zch.zipCode WHERE 1=1     AND acp.avgAssaultPerCa
pita > (         SELECT AVG(avgAssaultPerCapita)         FROM AssaultCrimePerCapita     )     AND zch.totalHappinessScore < (         SELECT AVG(totalH
appinessScore)         FROM ZipCodeHappiness     );
+---------+--------------------+--------------------+
| zipCode | avgAssaultPerCapita | totalHappinessScore |
+---------+--------------------+--------------------+
| 60623   |             0.0029 |              57.03 |
| 60649   |             0.0040 |              59.38 |
| 60644   |             0.0039 |              57.94 |
| 60628   |             0.0033 |              57.79 |
| 60621   |             0.0068 |              58.28 |
| 60619   |             0.0042 |              58.52 |
| 60636   |             0.0052 |              56.56 |
| 60612   |             0.0041 |              60.13 |
| 60624   |             0.0049 |              57.96 |
| 60637   |             0.0040 |              58.40 |
| 60620   |             0.0042 |              57.86 |
| 60617   |             0.0031 |              57.06 |
+---------+--------------------+--------------------+
12 rows in set (0.03 sec)
```

Note! Output is less than 15 rows.

Complex query #4

```
-- Active users for further research:
-- Select users that have taken the surveys in the most different zip code areas.
Order them in descending order by the number of different zip code areas.

SELECT u.email, u.firstName, u.lastName, COUNT(DISTINCT hs.zipCode) AS zip_code_count
FROM Users u
JOIN HappinessSurveys hs ON u.email = hs.userEmail
GROUP BY u.email, u.firstName, u.lastName
ORDER BY zip_code_count DESC;
```

```
mysql> SELECT u.email, u.firstName, u.lastName, COUNT(DISTINCT hs.zipCode) AS zip_code_count FROM Users u JOIN HappinessSurveys hs ON u.email = hs.user
Email GROUP BY u.email, u.firstName, u.lastName ORDER BY zip_code_count DESC LIMIT 15;
+---------------------+-----------+-----------+----------------+
| email               | firstName | lastName  | zip_code_count |
+---------------------+-----------+-----------+----------------+
| user1@example.com   | Richard   | Smith     |              1 |
| user10@example.com  | Michelle  | Martin    |              1 |
| user100@example.com | George    | Perez     |              1 |
| user1000@example.com | Ashley   | Wright    |              1 |
| user101@example.com | Sarah     | Torres    |              1 |
| user102@example.com | Matthew   | Scott     |              1 |
| user103@example.com | Richard   | Williams  |              1 |
| user104@example.com | Patricia  | Martin    |              1 |
| user105@example.com | Richard   | Hernandez |              1 |
| user106@example.com | Michelle  | Young     |              1 |
| user107@example.com | Matthew   | Wright    |              1 |
| user108@example.com | Paul      | Nguyen    |              1 |
| user109@example.com | Charles   | Flores    |              1 |
| user11@example.com  | Kenneth   | White     |              1 |
| user110@example.com | William   | Johnson   |              1 |
+---------------------+-----------+-----------+----------------+
```

Indexing

Complex query #1

Indexing configuration	Query Cost
No additional indexing	34,939.69
+ ZipAreas.stateCode	5,432.19
+ ZipAreas.city	4,589.76
+ ZipAreas.stateCode & ZipAreas.city	2,695.96

No additional indexing

EXPLAIN

-> Nested loop inner join (cost=34939.69 rows=0) (actual time=78.173..97.904 rows=1001 loops=1) -> Nested loop inner join (cost=28828.39 rows=2438) (actual time=30.931..48.711 rows=1130 loops=1) -> Filter: (chs.totalHappinessScore is not null) (cost=2461.75 rows=24375) (actual time=0.043..8.163 rows=24062 loops=1) -> Table scan on chs (cost=2461.75 rows=24375) (actual time=0.043..6.461 rows=24062 loops=1) -> Filter: ((za.stateCode = 'IL') and (za.city is not null)) (cost=0.98 rows=0.1) (actual time=0.002..0.002 rows=0 loops=24062) -> Single-row index lookup on za using PRIMARY (zipCode=chs.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=24062) -> Covering index lookup on maxScores using <auto_key0> (city=za.city, maxHappiness=chs.totalHappinessScore) (actual time=0.043..0.043 rows=1 loops=1130) -> Materialize (cost=0.00..0.00 rows=0) (actual time=47.231..47.231 rows=1001 loops=1) -> Table scan on <temporary> (actual time=46.065..46.220 rows=1001 loops=1) -> Aggregate using temporary table (actual time=46.061..46.061 rows=1001 loops=1) -> Nested loop inner join (cost=28828.39 rows=2438) (actual time=28.127..45.075 rows=1130 loops=1) -> Table scan on chs (cost=2461.75 rows=24375) (actual time=0.032..6.639 rows=24062 loops=1) -> Filter: (za.stateCode = 'IL') (cost=0.98 rows=0.1) (actual time=0.001..0.001 rows=0 loops=24062) -> Single-row index lookup on za using PRIMARY (zipCode=chs.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=24062)

+ ZipAreas.stateCode

EXPLAIN

-> Nested loop inner join (cost=5432.19 rows=0) (actual time=11.878..22.219 rows=1001 loops=1) -> Nested loop inner join (cost=1977.19 rows=1381) (actual time=0.642..8.757 rows=1130 loops=1) -> Filter: (za.city is not null) (cost=1493.84 rows=1381) (actual time=0.624..3.762 rows=1381 loops=1) -> Index lookup on za using za_stateCode_idx (stateCode='IL'), with index condition: (za.stateCode = 'IL') (cost=1493.84 rows=1381) (actual time=0.622..3.658 rows=1381 loops=1) -> Filter: (chs.totalHappinessScore is not null) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=1381) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=1381) -> Covering index lookup on maxScores using <auto_key0> (city=za.city, maxHappiness=chs.totalHappinessScore) (actual time=0.011..0.012 rows=1 loops=1130) -> Materialize (cost=0.00..0.00 rows=0) (actual time=11.227..11.227 rows=1001 loops=1) -> Table scan on <temporary> (actual time=10.156..10.294 rows=1001 loops=1) -> Aggregate using temporary table (actual time=10.154..10.154 rows=1001 loops=1) -> Nested loop inner join (cost=1977.19 rows=1381) (actual time=0.345..9.083 rows=1130 loops=1) -> Index lookup on za using za_stateCode_idx (stateCode='IL'), with index condition: (za.stateCode = 'IL') (cost=1493.84 rows=1381) (actual time=0.338..4.345 rows=1381 loops=1) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=1381)

+ ZipAreas.city

EXPLAIN

-> Nested loop inner join (cost=4589.76 rows=39) (actual time=50.547..101.726 rows=1001 loops=1) -> Nested loop inner join (cost=4454.58 rows=386) (actual time=50.521..88.885 rows=1136 loops=1) -> Filter: (maxScores.city is not null) (cost=0.11..276.66 rows=2437) (actual time=49.686..50.139 rows=1001 loops=1) -> Table scan on maxScores (cost=2.50..2.50 rows=0) (actual time=49.684..50.018 rows=1001 loops=1) -> Materialize (cost=0.00..0.00 rows=0) (actual time=49.683..49.683 rows=1001 loops=1) -> Table scan on <temporary> (actual time=49.301..49.441 rows=1001 loops=1) -> Aggregate using temporary table (actual time=49.297..49.297 rows=1001 loops=1) -> Nested loop inner join (cost=28828.39 rows=2438) (actual time=30.657..48.294 rows=1130 loops=1) -> Table scan on chs (cost=2461.75 rows=24375) (actual time=0.044..6.771 rows=24062 loops=1) -> Filter: (za.stateCode = 'IL') (cost=0.98 rows=0.1) (actual time=0.002..0.002 rows=0 loops=24062) -> Single-row index lookup on za using PRIMARY (zipCode=chs.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=24062) -> Filter: (za.stateCode = 'IL') (cost=1.56 rows=0.2) (actual time=0.033..0.038 rows=1 loops=1001) -> Index lookup on za using za_city_idx (city=maxScores.city) (cost=1.56 rows=2) (actual time=0.021..0.037 rows=5 loops=1001) -> Filter: (chs.totalHappinessScore = maxScores.maxHappiness) (cost=0.25 rows=0.1) (actual time=0.011..0.011 rows=1 loops=1136) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.011..0.011 rows=1 loops=1136)

+ ZipAreas.stateCode & ZipAreas.city

Complex query #2

Indexing configuration	Query Cost
No additional indexing	3,936.64
+ CrimeEvents.Date	8,177.59
+ ZipAreas.stateCode	5,495.69
+ CrimeEvents.Date & ZipAreas.stateCode	9,736.64

No additional indexing

EXPLAIN

-> Table scan on <except temporary> (cost=3936.64..3936.64 rows=0) (actual time=97.140..97.141 rows=6 loops=1) -> Except materialize with deduplication (cost=3934.14..3934.14 rows=0) (actual time=97.137..97.137 rows=6 loops=1) -> Nested loop inner join (cost=27.54 rows=0) (actual time=4.867..4.887 rows=6 loops=1) -> Table scan on avgHappiness (cost=2.50..2.50 rows=0) (actual time=4.850..4.851 rows=6 loops=1) -> Materialize (cost=0.00..0.00 rows=0) (actual time=4.849..4.849 rows=6 loops=1) -> Filter: (avg(hs2.userReportedHappinessScore) > 8) (actual time=4.821..4.836 rows=6 loops=1) -> Table scan on <temporary> (actual time=4.813..4.820 rows=48 loops=1) -> Aggregate using temporary table (actual time=4.811..4.811 rows=48 loops=1) -> Nested loop inner join (cost=1184.21 rows=100) (actual time=0.209..4.723 rows=48 loops=1) -> Table scan on hs2 (cost=102.50 rows=1000) (actual time=0.134..0.583 rows=1000 loops=1) -> Filter: (za.stateCode = 'IL') (cost=0.98 rows=0.1) (actual time=0.004..0.004 rows=0 loops=1000) -> Single-row index lookup on za using PRIMARY (zipCode=hs2.zipCode) (cost=0.98 rows=1) (actual time=0.004..0.004 rows=1 loops=1000) -> Covering index lookup on hs using zipCode (zipCode=avgHappiness.zipCode) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=6) -> Group (no aggregates) (cost=3906.60 rows=19172) (actual time=0.881..92.093 rows=71 loops=1) -> Filter: (ce.`date` >= <cache>((curdate() - interval 30 day))) (cost=1989.43 rows=19172) (actual time=0.591..90.199 rows=12820 loops=1) -> Index scan on ce using ce_zipCode_idx (cost=1989.43 rows=57521) (actual time=0.289..86.150 rows=57832 loops=1)

+ CrimeEvents.Date

EXPLAIN

-> Table scan on <except temporary> (cost=8177.59..8177.59 rows=0) (actual time=101.047..101.049 rows=6 loops=1) -> Except materialize with deduplication (cost=8175.09..8175.09 rows=0) (actual time=101.044..101.044 rows=6 loops=1) -> Nested loop inner join (cost=27.54 rows=0) (actual time=3.079..3.096 rows=6 loops=1) -> Table scan on avgHappiness (cost=2.50..2.50 rows=0) (actual time=3.065..3.065 rows=6 loops=1) -> Materialize (cost=0.00..0.00 rows=0) (actual time=3.064..3.064 rows=6 loops=1) -> Filter: (avg(hs2.userReportedHappinessScore) > 8) (actual time=3.033..3.049 rows=6 loops=1) -> Table scan on <temporary> (actual time=3.026..3.034 rows=48 loops=1) -> Aggregate using temporary table (actual time=3.024..3.024 rows=48 loops=1) -> Nested loop inner join (cost=1184.21 rows=100) (actual time=0.091..2.954 rows=48 loops=1) -> Table scan on hs2 (cost=102.50 rows=1000) (actual time=0.048..0.411 rows=1000 loops=1) -> Filter: (za.stateCode = 'IL') (cost=0.98 rows=0.1) (actual time=0.002..0.002 rows=0 loops=1000) -> Single-row index lookup on za using PRIMARY (zipCode=hs2.zipCode) (cost=0.98 rows=1) (actual time=0.002..0.002 rows=1 loops=1000) -> Covering index lookup on hs using zipCode (zipCode=avgHappiness.zipCode) (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loops=6) -> Group (no aggregates) (cost=8147.55 rows=23232) (actual time=0.583..97.721 rows=71 loops=1) -> Filter: (ce.`date` >= <cache>((curdate() - interval 30 day))) (cost=5824.35 rows=23232) (actual time=0.401..95.412 rows=12820 loops=1) -> Index scan on ce using ce_zipCode_idx (cost=5824.35 rows=57521) (actual time=0.199..90.215 rows=57832 loops=1)

+ ZipAreas.stateCode

EXPLAIN

-> Table scan on <except temporary> (cost=5475.53..5495.69 rows=1415) (actual time=90.992..90.993 rows=6 loops=1) -> Except materialize with deduplication (cost=5475.52..5475.52 rows=1415) (actual time=90.989..90.989 rows=6 loops=1) -> Nested loop inner join (cost=1427.44 rows=1415) (actual time=4.572..4.592 rows=6 loops=1) -> Table scan on avgHappiness (cost=916.24..936.18 rows=1398) (actual time=4.564..4.565 rows=6 loops=1) -> Materialize (cost=916.22..916.22 rows=1398) (actual time=4.562..4.562 rows=6 loops=1) -> Filter: (avg(hs2.userReportedHappinessScore) > 8) (cost=776.45 rows=1398) (actual time=0.195..4.548 rows=6 loops=1) -> Group aggregate: avg(hs2.userReportedHappinessScore), avg(hs2.userReportedHappinessScore) (cost=776.45 rows=1398) (actual time=0.191..4.529 rows=48 loops=1) -> Nested loop inner join (cost=636.67 rows=1398) (actual time=0.170..4.486 rows=48 loops=1) -> Filter: (za.stateCode = 'IL') (cost=147.45 rows=1381) (actual time=0.045..0.731 rows=1381 loops=1) -> Covering index lookup on za using za_stateCode_idx (stateCode='IL') (cost=147.45 rows=1381) (actual time=0.044..0.496 rows=1381 loops=1) -> Index lookup on hs2 using zipCode (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=0 loops=1381) -> Covering index lookup on hs using zipCode (zipCode=avgHappiness.zipCode) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=6) -> Group (no aggregates) (cost=3906.60 rows=19172) (actual time=0.576..86.229 rows=71 loops=1) -> Filter: (ce.`date` >= <cache>((curdate() - interval 30 day))) (cost=1989.43 rows=19172) (actual time=0.381..84.395 rows=12820 loops=1) -> Index scan on ce using ce_zipCode_idx (cost=1989.43 rows=57521) (actual time=0.180..80.305 rows=57832 loops=1)

+ CrimeEvents.Date & ZipAreas.stateCode

EXPLAIN

-> Table scan on <except temporary> (cost=9716.48..9736.64 rows=1415) (actual time=89.551..89.552 rows=6 loops=1) -> Except materialize with deduplication (cost=9716.47..9716.47 rows=1415) (actual time=89.547..89.547 rows=6 loops=1) -> Nested loop inner join (cost=1427.44 rows=1415) (actual time=4.506..4.526 rows=6 loops=1) -> Table scan on avgHappiness (cost=916.24..936.18 rows=1398) (actual time=4.496..4.497 rows=6 loops=1) -> Materialize (cost=916.22..916.22 rows=1398) (actual time=4.494..4.494 rows=6 loops=1) -> Filter: (avg(hs2.userReportedHappinessScore) > 8) (cost=776.45 rows=1398) (actual time=0.178..4.478 rows=6 loops=1) -> Group aggregate: avg(hs2.userReportedHappinessScore), avg(hs2.userReportedHappinessScore) (cost=776.45 rows=1398) (actual time=0.174..4.459 rows=48 loops=1) -> Nested loop inner join (cost=636.67 rows=1398) (actual time=0.152..4.416 rows=48 loops=1) -> Filter: (za.stateCode = 'IL') (cost=147.45 rows=1381) (actual time=0.054..0.771 rows=1381 loops=1) -> Covering index lookup on za using za_stateCode_idx (stateCode='IL') (cost=147.45 rows=1381) (actual time=0.052..0.555 rows=1381 loops=1) -> Index lookup on hs2 using zipCode (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=0 loops=1381) -> Covering index lookup on hs using zipCode (zipCode=avgHappiness.zipCode) (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loops=6) -> Group (no aggregates) (cost=8147.55 rows=23232) (actual time=0.659..84.846 rows=71 loops=1) -> Filter: (ce.`date` >= <cache>((curdate() - interval 30 day))) (cost=5824.35 rows=23232) (actual time=0.438..82.928 rows=12820 loops=1) -> Index scan on ce using ce_zipCode_idx (cost=5824.35 rows=57521) (actual time=0.238..78.875 rows=57832 loops=1)

Complex query #3

Indexing configuration	Query Cost
No additional indexing	83.62
+ CrimeEvents.primaryType	114.21
+ ZipAreas.StateCode	120.80
+ ZipAreas.City	64.68

No additional indexing

EXPLAIN

-> Nested loop inner join (cost=83.62 rows=2) (actual time=81.907..82.020 rows=12 loops=1) -> Nested loop inner join (cost=49.13 rows=32) (actual time=81.889..81.976 rows=12 loops=1) -> Filter: ((acp.avgAssaultPerCapita > (select #4)) and (acp.zipCode is not null)) (cost=0.16..15.65 rows=96) (actual time=38.537..38.562 rows=15 loops=1) -> Table scan on acp (cost=2.50..2.50 rows=0) (actual time=38.481..38.493 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (cost=0.00..0.00 rows=0) (actual time=38.480..38.480 rows=56 loops=1) -> Table scan on <temporary> (actual time=38.417..38.427 rows=56 loops=1) -> Aggregate using temporary table (actual time=38.415..38.415 rows=56 loops=1) -> Nested loop inner join (cost=12046.44 rows=288) (actual time=0.192..35.370 rows=5341 loops=1) -> Filter: ((ce.primaryType = 'ASSAULT') and (ce.zipCode is not null)) (cost=5824.35 rows=5752) (actual time=0.171..24.469 rows=5423 loops=1) -> Table scan on ce (cost=5824.35 rows=57521) (actual time=0.074..17.245 rows=57832 loops=1) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.002..0.002 rows=1 loops=5423) -> Single-row index lookup on za using PRIMARY (zipCode=ce.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=5423) -> Select #4 (subquery in condition; run only once) -> Aggregate: avg(AssaultCrimePerCapita.avgAssaultPerCapita) (cost=2.50..2.50 rows=1) (actual time=0.024..0.025 rows=1 loops=1) -> Table scan on AssaultCrimePerCapita (cost=2.50..2.50 rows=0) (actual time=0.006..0.013 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (query plan printed elsewhere) (cost=0.00..0.00 rows=0) (never executed) -> Filter: (chs.totalHappinessScore < (select #6)) (cost=0.25 rows=0.3) (actual time=2.894..2.894 rows=1 loops=15) -> Single-row index lookup on chs using PRIMARY (zipCode=acp.zipCode) (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loops=15) -> Select #6 (subquery in condition; run only once) -> Aggregate: avg(chs.totalHappinessScore) (cost=28950.26 rows=1) (actual time=43.310..43.311 rows=1 loops=1) -> Nested loop inner join (cost=28828.39 rows=1219) (actual time=27.101..43.288 rows=54 loops=1) -> Table scan on chs (cost=2461.75 rows=24375) (actual time=0.024..6.324 rows=24062 loops=1) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.001..0.001 rows=0 loops=24062) -> Single-row index lookup on za using PRIMARY (zipCode=chs.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=24062) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.003..0.003 rows=1 loops=12) -> Single-row index lookup on za using PRIMARY (zipCode=acp.zipCode) (cost=0.98 rows=1) (actual time=0.003..0.003 rows=1 loops=12)

+ CrimeEvents.primaryType

EXPLAIN

-> Nested loop inner join (cost=114.21 rows=2) (actual time=29.233..29.326 rows=12 loops=1) -> Nested loop inner join (cost=112.62 rows=5) (actual time=26.067..26.124 rows=15 loops=1) -> Filter: ((acp.avgAssaultPerCapita > (select #4)) and (acp.zipCode is not null)) (cost=0.17..14.92 rows=90) (actual time=26.056..26.077 rows=15 loops=1) -> Table scan on acp (cost=2.50..2.50 rows=0) (actual time=25.997..26.007 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (cost=0.00..0.00 rows=0) (actual time=25.996..25.996 rows=56 loops=1) -> Table scan on <temporary> (actual time=25.911..25.919 rows=56 loops=1) -> Aggregate using temporary table (actual time=25.908..25.908 rows=56 loops=1) -> Nested loop inner join (cost=6625.15 rows=271) (actual time=0.329..22.565 rows=5341 loops=1) -> Filter: (ce.zipCode is not null) (cost=759.05 rows=5423) (actual time=0.309..10.336 rows=5423 loops=1) -> Index lookup on ce using ce_primaryType_idx (primaryType='ASSAULT') (cost=759.05 rows=5423) (actual time=0.307..9.853 rows=5423 loops=1) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.002..0.002 rows=1 loops=5423) -> Single-row index lookup on za using PRIMARY (zipCode=ce.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.002 rows=1 loops=5423) -> Select #4 (subquery in condition; run only once) -> Aggregate: avg(AssaultCrimePerCapita.avgAssaultPerCapita) (cost=2.50..2.50 rows=1) (actual time=0.028..0.028 rows=1 loops=1) -> Table scan on AssaultCrimePerCapita (cost=2.50..2.50 rows=0) (actual time=0.006..0.012 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (query plan printed elsewhere) (cost=0.00..0.00 rows=0) (never executed) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.003..0.003 rows=1 loops=15) -> Single-row index lookup on za using PRIMARY (zipCode=acp.zipCode) (cost=0.98 rows=1) (actual time=0.002..0.002 rows=1 loops=15) -> Filter: (chs.totalHappinessScore < (select #6)) (cost=0.26 rows=0.3) (actual time=0.213..0.213 rows=1 loops=15) -> Single-row index lookup on chs using PRIMARY (zipCode=acp.zipCode) (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=15) -> Select #6 (subquery in condition; run only once) -> Aggregate: avg(chs.totalHappinessScore) (cost=1431.69 rows=1) (actual time=3.133..3.134 rows=1 loops=1) -> Nested loop inner join (cost=1417.88 rows=138) (actual time=0.720..3.120 rows=54 loops=1) -> Filter: (za.city = 'Chicago') (cost=1369.55 rows=138) (actual time=0.713..3.008 rows=56 loops=1) -> Index lookup on za using za_stateCode_idx (stateCode='IL'), with index condition: (za.stateCode = 'IL') (cost=1369.55 rows=1381) (actual time=0.259..2.872 rows=1381 loops=1) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=56)

+ ZipAreas.StateCode

EXPLAIN

-> Nested loop inner join (cost=120.80 rows=2) (actual time=45.029..45.139 rows=12 loops=1) -> Nested loop inner join (cost=119.13 rows=5) (actual time=41.958..42.028 rows=15 loops=1) -> Filter: ((acp.avgAssaultPerCapita > (select #4)) and (acp.zipCode is not null)) (cost=0.16..15.65 rows=96) (actual time=41.947..41.969 rows=15 loops=1) -> Table scan on acp (cost=2.50..2.50 rows=0) (actual time=41.897..41.907 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (cost=0.00..0.00 rows=0) (actual time=41.896..41.896 rows=56 loops=1) -> Table scan on <temporary> (actual time=41.840..41.850 rows=56 loops=1) -> Aggregate using temporary table (actual time=41.838..41.838 rows=56 loops=1) -> Nested loop inner join (cost=12046.44 rows=288) (actual time=0.134..38.517 rows=5341 loops=1) -> Filter: ((ce.primaryType = 'ASSAULT') and (ce.zipCode is not null)) (cost=5824.35 rows=5752) (actual time=0.121..26.886 rows=5423 loops=1) -> Table scan on ce (cost=5824.35 rows=57521) (actual time=0.034..19.249 rows=57832 loops=1) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.002..0.002 rows=1 loops=5423) -> Single-row index lookup on za using PRIMARY (zipCode=ce.zipCode) (cost=0.98 rows=1) (actual time=0.001..0.001 rows=1 loops=5423) -> Select #4 (subquery in condition; run only once) -> Aggregate: avg(AssaultCrimePerCapita.avgAssaultPerCapita) (cost=2.50..2.50 rows=1) (actual time=0.022..0.022 rows=1 loops=1) -> Table scan on AssaultCrimePerCapita (cost=2.50..2.50 rows=0) (actual time=0.006..0.012 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (query plan printed elsewhere) (cost=0.00..0.00 rows=0) (never executed) -> Filter: ((za.city = 'Chicago') and (za.stateCode = 'IL')) (cost=0.98 rows=0.05) (actual time=0.004..0.004 rows=1 loops=15) -> Single-row index lookup on za using PRIMARY (zipCode=acp.zipCode) (cost=0.98 rows=1) (actual time=0.003..0.003 rows=1 loops=15) -> Filter: (chs.totalHappinessScore < (select #6)) (cost=0.26 rows=0.3) (actual time=0.207..0.207 rows=1 loops=15) -> Single-row index lookup on chs using PRIMARY (zipCode=acp.zipCode) (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=15) -> Select #6 (subquery in condition; run only once) -> Aggregate: avg(chs.totalHappinessScore) (cost=1431.69 rows=1) (actual time=3.042..3.043 rows=1 loops=1) -> Nested loop inner join (cost=1417.88 rows=138) (actual time=0.678..3.030 rows=54 loops=1) -> Filter: (za.city = 'Chicago') (cost=1369.55 rows=138) (actual time=0.670..2.903 rows=56 loops=1) -> Index lookup on za using za_stateCode_idx (stateCode='IL'), with index condition: (za.stateCode = 'IL') (cost=1369.55 rows=1381) (actual time=0.236..2.765 rows=1381 loops=1) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=56)

+ ZipAreas.City

EXPLAIN

-> Nested loop inner join (cost=64.68 rows=7) (actual time=164.730..164.982 rows=12 loops=1) -> Nested loop inner join (cost=57.50 rows=2) (actual time=0.831..1.052 rows=31 loops=1) -> Filter: (za.stateCode = 'IL') (cost=55.54 rows=6) (actual time=0.327..0.364 rows=56 loops=1) -> Index lookup on za using za_city_idx (city='Chicago') (cost=55.54 rows=56) (actual time=0.323..0.344 rows=56 loops=1) -> Filter: (chs.totalHappinessScore < (select #6)) (cost=0.26 rows=0.3) (actual time=0.012..0.012 rows=1 loops=56) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.26 rows=1) (actual time=0.004..0.004 rows=1 loops=56) -> Select #6 (subquery in condition; run only once) -> Aggregate: avg(chs.totalHappinessScore) (cost=58.06 rows=1) (actual time=0.420..0.420 rows=1 loops=1) -> Nested loop inner join (cost=57.50 rows=6) (actual time=0.192..0.400 rows=54 loops=1) -> Filter: (za.stateCode = 'IL') (cost=55.54 rows=6) (actual time=0.182..0.212 rows=56 loops=1) -> Index lookup on za using za_city_idx (city='Chicago') (cost=55.54 rows=56) (actual time=0.181..0.195 rows=56 loops=1) -> Single-row index lookup on chs using PRIMARY (zipCode=za.zipCode) (cost=0.27 rows=1) (actual time=0.003..0.003 rows=1 loops=56) -> Filter: (acp.avgAssaultPerCapita > (select #4)) (cost=0.80..2.95 rows=4) (actual time=5.288..5.288 rows=0 loops=31) -> Covering index lookup on acp using <auto_key0> (zipCode=za.zipCode) (actual time=5.285..5.285 rows=1 loops=31) -> Materialize CTE AssaultCrimePerCapita if needed (cost=0.00..0.00 rows=0) (actual time=163.771..163.771 rows=56 loops=1) -> Table scan on <temporary> (actual time=163.557..163.580 rows=56 loops=1) -> Aggregate using temporary table (actual time=163.554..163.554 rows=56 loops=1) -> Nested loop inner join (cost=1482.64 rows=408) (actual time=0.525..157.891 rows=5341 loops=1) -> Filter: (za.stateCode = 'IL') (cost=55.54 rows=6) (actual time=0.186..0.332 rows=56 loops=1) -> Index lookup on za using za_city_idx (city='Chicago') (cost=55.54 rows=56) (actual time=0.185..0.285 rows=56 loops=1) -> Filter: (ce.primaryType = 'ASSAULT') (cost=183.33 rows=73) (actual time=0.255..2.804 rows=95 loops=56) -> Index lookup on ce using ce_zipCode_idx (zipCode=za.zipCode) (cost=183.33 rows=728) (actual time=0.248..2.595 rows=1018 loops=56) -> Select #4 (subquery in condition; run only once) -> Aggregate: avg(AssaultCrimePerCapita.avgAssaultPerCapita) (cost=2.50..2.50 rows=1) (actual time=0.034..0.034 rows=1 loops=1) -> Table scan on AssaultCrimePerCapita (cost=2.50..2.50 rows=0) (actual time=0.007..0.017 rows=56 loops=1) -> Materialize CTE AssaultCrimePerCapita if needed (query plan printed elsewhere) (cost=0.00..0.00 rows=0) (never executed)

Complex query #4

Indexing configuration	Query Cost
No additional indexing	541.45
+ User.lastName	541.45
+ User.firstName	541.45
+ User.lastName & User.firstName	541.45

Note!
No performance change is observed with these indexes. This is likely due to the fact that the query is already grouping by the primary key of the Users table and, thus, the DBMS most likely already enhances the query and ignores the subsequent group by attributes of the Users table. The query has only primary keys left as attributes, and no further indexing configurations can be tried.

No additional indexing

EXPLAIN

-> Sort: zip_code_count DESC (actual time=7.206..7.295 rows=1000 loops=1) -> Stream results (cost=541.45 rows=974) (actual time=0.838..6.976 rows=1000 loops=1) -> Group aggregate: count(distinct hs.zipCode) (cost=541.45 rows=974) (actual time=0.835..6.451 rows=1000 loops=1) -> Nested loop inner join (cost=444.05 rows=974) (actual time=0.820..5.728 rows=1000 loops=1) -> Sort: u.email, u.firstName, u.lastName (cost=103.15 rows=974) (actual time=0.797..0.897 rows=1000 loops=1) -> Table scan on u (cost=103.15 rows=974) (actual time=0.042..0.414 rows=1000 loops=1) -> Covering index lookup on hs using PRIMARY (userEmail=u.email) (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loops=1000)

+ User.lastName

EXPLAIN

-> Sort: zip_code_count DESC (actual time=7.249..7.341 rows=1000 loops=1) -> Stream results (cost=541.45 rows=974) (actual time=0.895..6.886 rows=1000 loops=1) -> Group aggregate: count(distinct hs.zipCode) (cost=541.45 rows=974) (actual time=0.892..6.363 rows=1000 loops=1) -> Nested loop inner join (cost=444.05 rows=974) (actual time=0.878..5.633 rows=1000 loops=1) -> Sort: u.email, u.firstName, u.lastName (cost=103.15 rows=974) (actual time=0.853..0.951 rows=1000 loops=1) -> Table scan on u (cost=103.15 rows=974) (actual time=0.054..0.477 rows=1000 loops=1) -> Covering index lookup on hs using PRIMARY (userEmail=u.email) (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loops=1000)

+ User.firstName

EXPLAIN

-> Sort: zip_code_count DESC (actual time=7.249..7.341 rows=1000 loops=1) -> Stream results (cost=541.45 rows=974) (actual time=0.895..6.886 rows=1000 loops=1) -> Group aggregate: count(distinct hs.zipCode) (cost=541.45 rows=974) (actual time=0.892..6.363 rows=1000 loops=1) -> Nested loop inner join (cost=444.05 rows=974) (actual time=0.878..5.633 rows=1000 loops=1) -> Sort: u.email, u.firstName, u.lastName (cost=103.15 rows=974) (actual time=0.853..0.951 rows=1000 loops=1) -> Table scan on u (cost=103.15 rows=974) (actual time=0.054..0.477 rows=1000 loops=1) -> Covering index lookup on hs using PRIMARY (userEmail=u.email) (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loops=1000)

+ User.lastName & User.firstName

EXPLAIN

-> Sort: zip_code_count DESC (actual time=7.249..7.341 rows=1000 loops=1) -> Stream results (cost=541.45 rows=974) (actual time=0.895..6.886 rows=1000 loops=1) -> Group aggregate: count(distinct hs.zipCode) (cost=541.45 rows=974) (actual time=0.892..6.363 rows=1000 loops=1) -> Nested loop inner join (cost=444.05 rows=974) (actual time=0.878..5.633 rows=1000 loops=1) -> Sort: u.email, u.firstName, u.lastName (cost=103.15 rows=974) (actual time=0.853..0.951 rows=1000 loops=1) -> Table scan on u (cost=103.15 rows=974) (actual time=0.054..0.477 rows=1000 loops=1) -> Covering index lookup on hs using PRIMARY (userEmail=u.email) (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loops=1000)

Final Indexing Configuration

The final indexing configuration that we have chosen for our project is the addition of indexes for attributes ZipAreas.stateCode and ZipAreas.city. Based on the results, we see a very significant performance improvement in Complex Query #1 when using both of these attributes as indexes. A small improvement was also observed in Complex Query #3 when the ZipAreas.city was indexed, which further justified this decision.

Although with Complex Query #2 we observed a slight increase in cost when ZipAreas.stateCode was used as an index, we concluded that the benefits observed with Complex Query #1 of using both attributes as index outweigh the increased cost in Complex Query #2. We suspect that Complex Query #1 will be used more often than Complex Query #2.