TEAM 106 - DATA BUDS
STAGE 3
Database Implementation and Indexing

Part 1: Data Definition Language (DDL) Commands and Queries

Courses

```sql
CREATE TABLE Courses (
    CourseId INT PRIMARY KEY,
    Topic VARCHAR(100) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM Courses;
+----------+
| COUNT(*) |
+----------+
|      433 |
+----------+
1 row in set (0.00 sec)
```

User

```sql
CREATE TABLE User (
    UserId INT  PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    Password VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM User;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.00 sec)
```

Tutor

```sql
CREATE TABLE Tutor (
    TutorId INT PRIMARY KEY,
    Topic VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT COUNT(*) FROM Tutor;
+----------+
| COUNT(*) |
+----------+
|     1000 |
+----------+
1 row in set (0.01 sec)
```

Problems:

```sql
CREATE TABLE Problems (
    Questions VARCHAR(50) PRIMARY KEY,
    Answers VARCHAR(50) NOT NULL,
    Difficulty INT,
    Topic VARCHAR(50) NOT NULL
);
```

```
mysql> SELECT Count(*) From Problems;
+----------+
| Count(*) |
+----------+
|    44515 |
+----------+
1 row in set (0.00 sec)
```

Info:

```sql
CREATE TABLE Info (
    InfoID INT PRIMARY KEY ,
    Topic varchar(50) NOT NULL,
    WebURL varchar(250),
    ResourceType varchar(50)
);
```

```
mysql> SELECT Count(*) FROM Info;
+----------+
| Count(*) |
+----------+
|       65 |
+----------+
1 row in set (0.01 sec)
```

Advanced Queries:

```
mysql> SELECT C.Topic, AVG(P.Difficulty) AS AvgDifficulty
    -> FROM Courses C
    -> JOIN Problems P ON C.Topic = P.Topic
    -> GROUP BY C.Topic
    -> LIMIT 15;
+-----------------------------------------------------------------+---------------+
| Topic                                                           | AvgDifficulty |
+-----------------------------------------------------------------+---------------+
| Advanced Topics in Natural Language Processing                  |        6.6071 |
| Advanced Topics in Security, Privacy, and Machine Learning      |        5.2308 |
| Applied Machine Learning                                        |        5.5588 |
| Artificial Intelligence                                         |        4.9310 |
| Computer Vision                                                 |        4.5758 |
| Data Science Discovery                                          |        5.9737 |
| Deep Learning                                                   |        5.7333 |
| Deep Learning for Computer Vision                               |        5.1071 |
| Introduction to Computer Science II                             |          NULL |
| Introduction to Data Mining                                     |        5.6154 |
| Machine Learning                                                |        6.0800 |
| Machine Learning for Bioinformatics                             |        5.7500 |
| Machine Learning for Signal Processing                          |        6.1143 |
| Modeling and Learning in Data Science                           |        5.2647 |
| Natural Language Processing                                     |        5.3030 |
+-----------------------------------------------------------------+---------------+
15 rows in set (1.00 sec)
```

```
mysql> Select Distinct c.Topic, Count(q.Question) From Courses c Join Problems q on c.Topic = q.Topic Where q.Question Like '%A*%' Group By c.Topic Limit 15;
+-------------------------------------------------+----------------------+
| Topic                                           | Count(q.Question) |
+-------------------------------------------------+----------------------+
| Computer Vision                                 |                    2 |
| Data Science Discovery                          |                   38 |
| Machine Learning                                |                    2 |
| Applied Machine Learning                        |                    5 |
| Advanced Topics in Natural Language Processing |                    1 |
| Introduction to Computer Science II             |                   17 |
+-------------------------------------------------+----------------------+
6 rows in set (0.04 sec)
```

```
mysql> SELECT I.WebURL FROM Info I natural join (SELECT C.Topic FROM Courses C natural join Problems P  GROUP BY C.Topic HAVING ROUND(AVG(P.Difficulty), 3) = 4.931) AS AvgDiffTopics;

+-----------------------------------------------------------------------------------------------------------------------+
| WebURL                                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------+
| https://www.youtube.com/watch?v=XXqr0cPQMZA&ab_channel=Staffbase                                                      |
| https://www.youtube.com/watch?v=Yq0QkCxoTHM                                                                           |
| https://www.analyticsvidhya.com/blog/2021/06/a-beginners-guide-to-artificial-intelligence/                            |
| https://towardsdatascience.com/the-future-of-artificial-intelligence-ff64f4a5bce7                                     |
| https://builtin.com/artificial-intelligence                                                                           |
| https://www.forbes.com/sites/bernardmarr/2020/07/06/the-top-5-business-benefits-of-artificial-intelligence-ai/ |
| https://www.ibm.com/cloud/learn/what-is-artificial-intelligence                                                       |
| https://www.sas.com/en_us/insights/analytics/what-is-artificial-intelligence.html                                     |
| https://www.microsoft.com/en-us/ai/ai-lab                                                                             |
| https://www.nvidia.com/en-us/deep-learning-ai/what-is-deep-learning/                                                  |
| https://www.techradar.com/news/what-is-ai-artificial-intelligence-explained                                           |
| https://www.mckinsey.com/featured-insights/artificial-intelligence                                                    |
| https://www.technologyreview.com/2021/10/27/1038498/artificial-intelligence-explained/                                |
| https://www.datacamp.com/community/blog/what-is-artificial-intelligence                                               |
| https://hbr.org/2019/07/artificial-intelligence-for-the-real-world                                                    |
+-----------------------------------------------------------------------------------------------------------------------+
15 rows in set (0.90 sec)
```

```
mysql> SELECT DISTINCT T.TutorId FROM Tutor T natural join Problems P  JOIN (SELECT C.Topic, ROUND(AVG(P.Difficulty), 3) AS AvgDifficulty FROM Courses C JOIN Problems P ON C.Topic = P.Topic GROUP
 BY C.Topic HAVING AvgDifficulty = 4.9310) AS AvgDiffTopics ON P.Topic = AvgDiffTopics.Topic;
+---------+
| TutorId |
+---------+
|     104 |
|     384 |
|     900 |
+---------+
3 rows in set (0.90 sec)
```

Indexing:

Indexing for Query:

Select C.Topic, AVG(P.Difficulty) AS AVGDifficulty FROM Courses C Join Problems P on C.topic = P.Topic Group By C.Topic

```
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=2.156..2.157 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=2.155..2.155 rows=15 loops=1)
        -> Filter: (P.Topic = C.Topic)  (cost=19531.96 rows=19485) (actual
time=0.196..1.156 rows=2303 loops=1)
            -> Inner hash join (<hash>(P.Topic)=<hash>(C.Topic))  (cost=19531.96
rows=19485) (actual time=0.194..0.513 rows=2303 loops=1)
                -> Table scan on P  (cost=0.02 rows=450) (actual time=0.027..0.125 rows=450
loops=1)
                -> Hash
                    -> Table scan on C  (cost=44.05 rows=433) (actual time=0.032..0.095
rows=433 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Indexes attempted
CREATE INDEX question_topic ON Courses(Topic);
CREATE INDEX question_topic2 ON Problems(Topic);
Create Index difficulty ON Problems(Difficulty);

I started out by trying a similar tactic to the last query by first creating an index for the Courses topic in order to reduce the cost of the join. This turned out to be very fruitful as it reduced the massive cost of 19531 for the join clause down to only 398. This likely meant that there were far more problems being pulled than in the previous query thus magnifying the impact of the cost saving “look up table” indexing

Once again I attempted to index on the Problems(Topic), but this led to no performance increase. Double indexing on the same operation seems to have no real benefit for the actual performance. As it is only performing one comparison from the smaller table to the larger table, rather than doing two comparisons in both directions. This makes sense as it would be very cost inefficient to do some form of dual checking.

Lastly I attempted to index on the difficulty of the problems which turned out to have a small positive impact on time but had no impact on cost. My only rationale is that while it might not improve the cost as all difficulties have to be accessed and scanned anyway, it could be attributed to faster lookups of specific difficulty values. Alternatively it might just be a thing with google cloud sometimes being faster or slower. Since it did not have a negative impact, I included it.

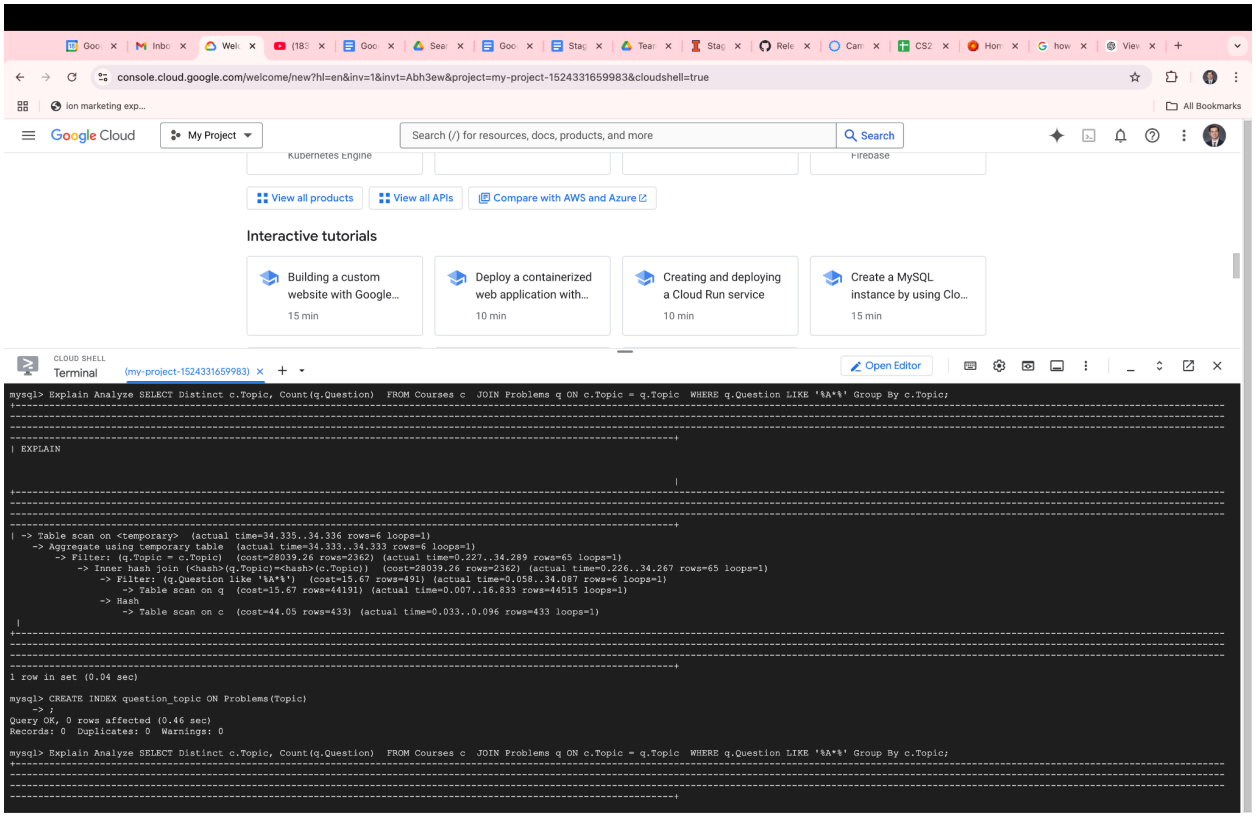
Final Index:
CREATE INDEX question_topic ON Courses(Topic);
CREATE INDEX question_topic3 ON Problems(Difficulty);

mysql> CREATE INDEX question_topic ON Courses(Topic);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> CREATE INDEX question_topic3 ON Problems(Difficulty);
Query OK, 0 rows affected (0.15 sec)
Records: 0 Duplicates: 0 Warnings: 0

```
mysql> CREATE INDEX question_topic ON Courses(Topic);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain Analyze Select C.Topic, AVG(P.Difficulty) AS AVGDifficulty FROM Courses C Join AI_Questions P on C.topic = P.Topic Group By C.Topic
    -> ;
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
| EXPLAIN
                                                                                |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=3.136..3.138 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=3.135..3.135 rows=15 loops=1)
        -> Nested loop inner join  (cost=398.89 rows=2189) (actual time=0.064..2.098 rows=2303 loops=1)
            -> Filter: (P.Topic is not null)  (cost=46.75 rows=450) (actual time=0.044..0.164 rows=450 loops=1)
                -> Table scan on P  (cost=46.75 rows=450) (actual time=0.043..0.140 rows=450 loops=1)
            -> Covering index lookup on C using question_topic (Topic=P.Topic)  (cost=0.30 rows=5) (actual time=0.002..0.004 rows=5 loops=450)
 |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> CREATE INDEX question_topic2 ON Problems(Topic);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain Analyze Select C.Topic, AVG(P.Difficulty) AS AVGDifficulty FROM Courses C Join AI_Questions P on C.topic = P.Topic Group By C.Topic
    -> ;
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
| EXPLAIN
                                                                                |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=4.838..4.840 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=4.837..4.837 rows=15 loops=1)
        -> Nested loop inner join  (cost=398.89 rows=2189) (actual time=0.078..3.145 rows=2303 loops=1)
            -> Filter: (P.Topic is not null)  (cost=46.75 rows=450) (actual time=0.058..0.236 rows=450 loops=1)
                -> Table scan on P  (cost=46.75 rows=450) (actual time=0.057..0.202 rows=450 loops=1)
            -> Covering index lookup on C using question_topic (Topic=P.Topic)  (cost=0.30 rows=5) (actual time=0.003..0.006 rows=5 loops=450)
 |
+--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Indexing for Query:

SELECT Distinct c.Topic, Count(q.Question) FROM Courses c JOIN Problems q ON c.Topic = q.Topic WHERE q.Question LIKE '%A*%' Group By c.Topic Limit 15;

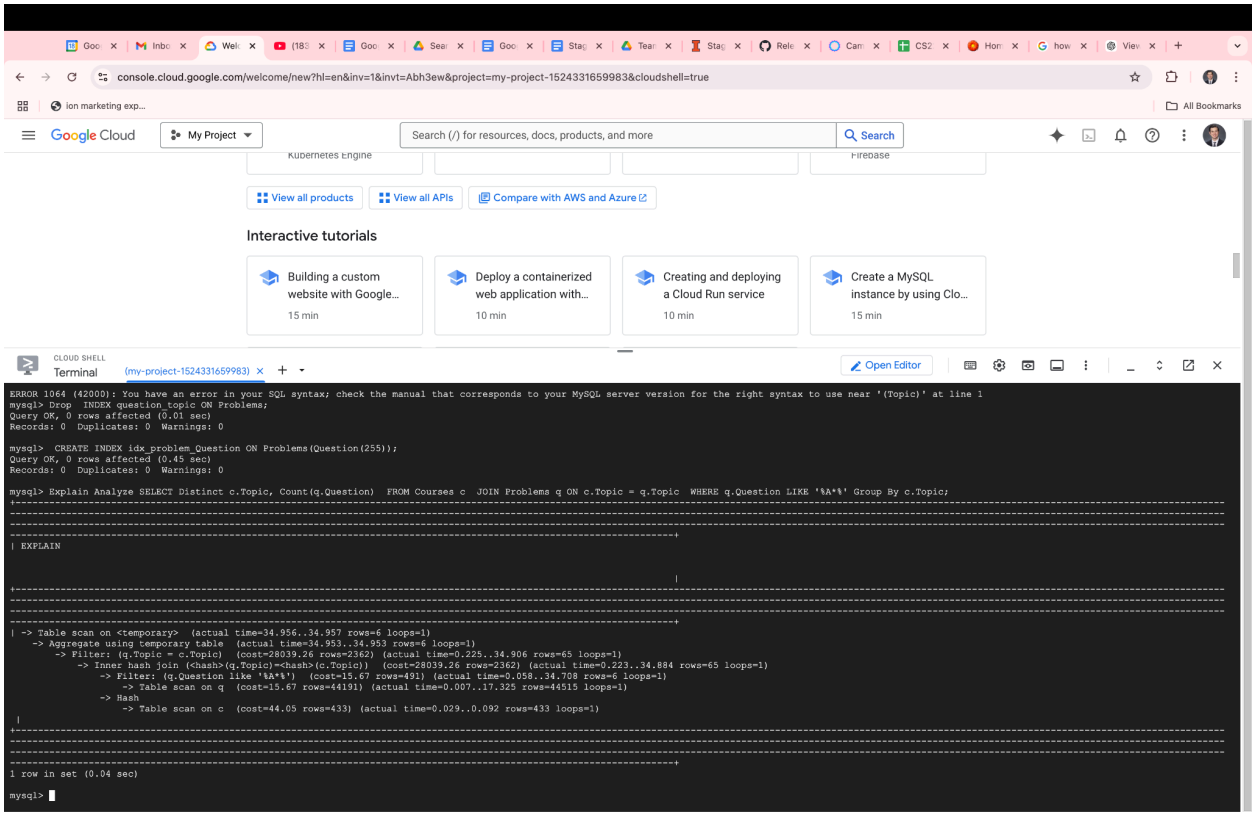


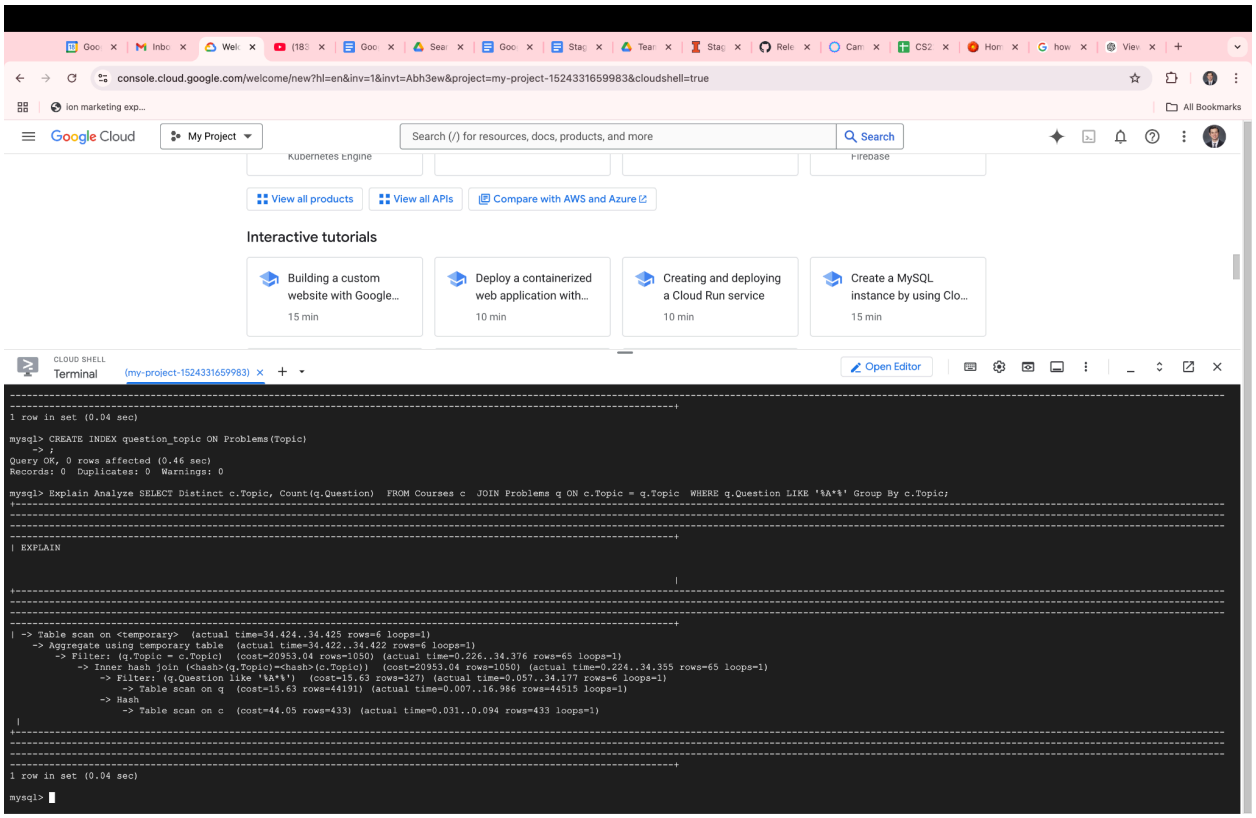
Indexes I tried:

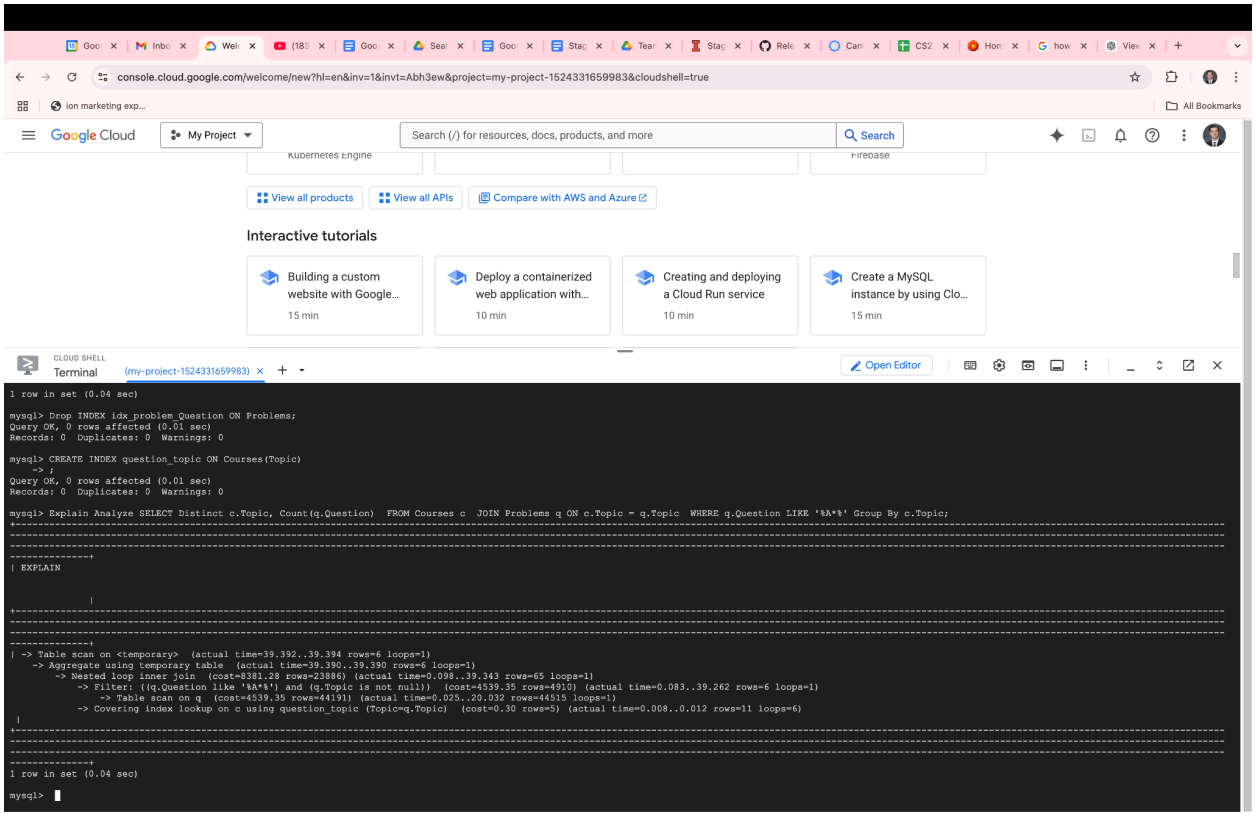
No Effect: CREATE INDEX idx_problem_Question ON Problems(Question(255));

Slightly Better: CREATE INDEX question_topic ON Problems(Topic)

Better: CREATE INDEX question_topic ON Courses(Topic)


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
View all products
View all APIs
Compare with AWS and Azure
Interactive tutorials
Building a custom website with Google...
15 min
Deploy a containerized web application with...
10 min
Creating and deploying a Cloud Run service
10 min
Create a MySQL instance by using Clo...
15 min
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '(Topic)' at line 1
mysql> Drop INDEX question_topic ON Problems;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> CREATE INDEX idx_problem_Question ON Problems(Question(255));
Query OK, 0 rows affected (0.45 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> Explain Analyze SELECT Distinct c.Topic, Count(q.Question) FROM Courses c JOIN Problems q ON c.Topic = q.Topic WHERE q.Question LIKE '%A*%' Group By c.Topic;
| EXPLAIN
| -> Table scan on <temporary> (actual time=34.956..34.957 rows=6 loops=1)
-> Aggregate using temporary table (actual time=34.953..34.953 rows=6 loops=1)
-> Filter: (q.Topic = c.Topic) (cost=28039.26 rows=2362) (actual time=0.225..34.906 rows=65 loops=1)
-> Inner hash join (<hash>(q.Topic)=<hash>(c.Topic)) (cost=28039.26 rows=2362) (actual time=0.223..34.884 rows=65 loops=1)
-> Filter: (q.Question like '%A*%') (cost=15.67 rows=491) (actual time=0.058..34.708 rows=6 loops=1)
-> Table scan on q (cost=15.67 rows=44191) (actual time=0.007..17.325 rows=44515 loops=1)
-> Hash
-> Table scan on c (cost=44.05 rows=433) (actual time=0.029..0.092 rows=433 loops=1)
1 row in set (0.04 sec)
mysql>


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
View all products
View all APIs
Compare with AWS and Azure
Interactive tutorials
Building a custom website with Google...
15 min
Deploy a containerized web application with...
10 min
Creating and deploying a Cloud Run service
10 min
Create a MySQL instance by using Clo...
15 min
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
1 row in set (0.04 sec)
mysql> CREATE INDEX question_topic ON Problems(Topic)
-> ;
Query OK, 0 rows affected (0.46 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> Explain Analyze SELECT Distinct c.Topic, Count(q.Question) FROM Courses c JOIN Problems q ON c.Topic = q.Topic WHERE q.Question LIKE '%A*%' Group By c.Topic;
| EXPLAIN
| -> Table scan on <temporary> (actual time=34.424..34.425 rows=6 loops=1)
-> Aggregate using temporary table (actual time=34.422..34.422 rows=6 loops=1)
-> Filter: (q.Topic = c.Topic) (cost=20953.04 rows=1050) (actual time=0.226..34.376 rows=65 loops=1)
-> Inner hash join (<hash>(q.Topic)=<hash>(c.Topic)) (cost=20953.04 rows=1050) (actual time=0.224..34.355 rows=65 loops=1)
-> Filter: (q.Question like '%A*%') (cost=15.63 rows=327) (actual time=0.057..34.177 rows=6 loops=1)
-> Table scan on q (cost=15.63 rows=44191) (actual time=0.007..16.986 rows=44515 loops=1)
-> Hash
-> Table scan on c (cost=44.05 rows=433) (actual time=0.031..0.094 rows=433 loops=1)
1 row in set (0.04 sec)
mysql>



Explanation:

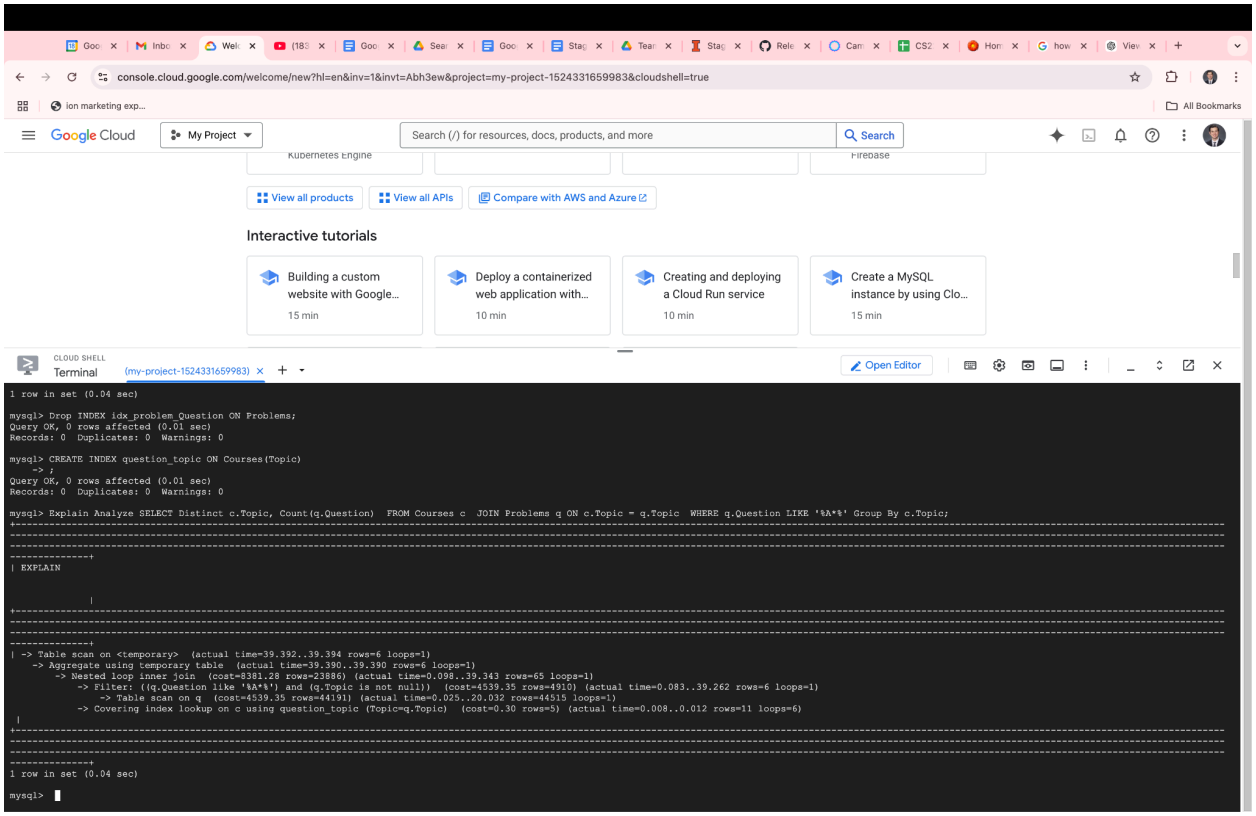
I tried a variety of different indexes for this particular query. I first attempted to Create an index for Problems(Question(255)). The logic behind this was to minimize the cost of performing the LIKE clause for where the questions is accessed because I thought the database would be able to quickly “look up” A*. This was an incorrect assumption though as it had no effect on the actual cost likely because the individual entries still have to be fully scanned every time to find whether there exists the substring A* anywhere in it, not just at the beginning or at a specific point in the string.

I then attempted to index on the Problems(Topic), but this only showed a small performance increase: Cost for filter decreased from 28039 to 20953. My hypothesis was that, since a large portion of the cost (in fact most of it) came from the join table clause, the cost would be heavily reduced by creating an index that the database would be able to immediately look up rather than having to scan the full tables every single time. This turned out to be false though, and when I examined the table sizes I realized that the problems table is many times larger than the Courses table. This likely means that rather than the database trying to join Problems to Courses; it was joining courses to problems in order to avoid excessive comparisons. This meant that indexing Courses rather than problems would likely save time, which is what I did next.

Finally I indexed Courses(topic) and there was an immediate performance improvement: the cost decreased from 28039 to 8381. This meant that by adding indexes to the Courses table the database was able to find matches faster. This change ended up being the only index that I added. When I attempted to add other indexes on top of this one, such as the previous two it only led to performance decreases, either in terms of cost or time.

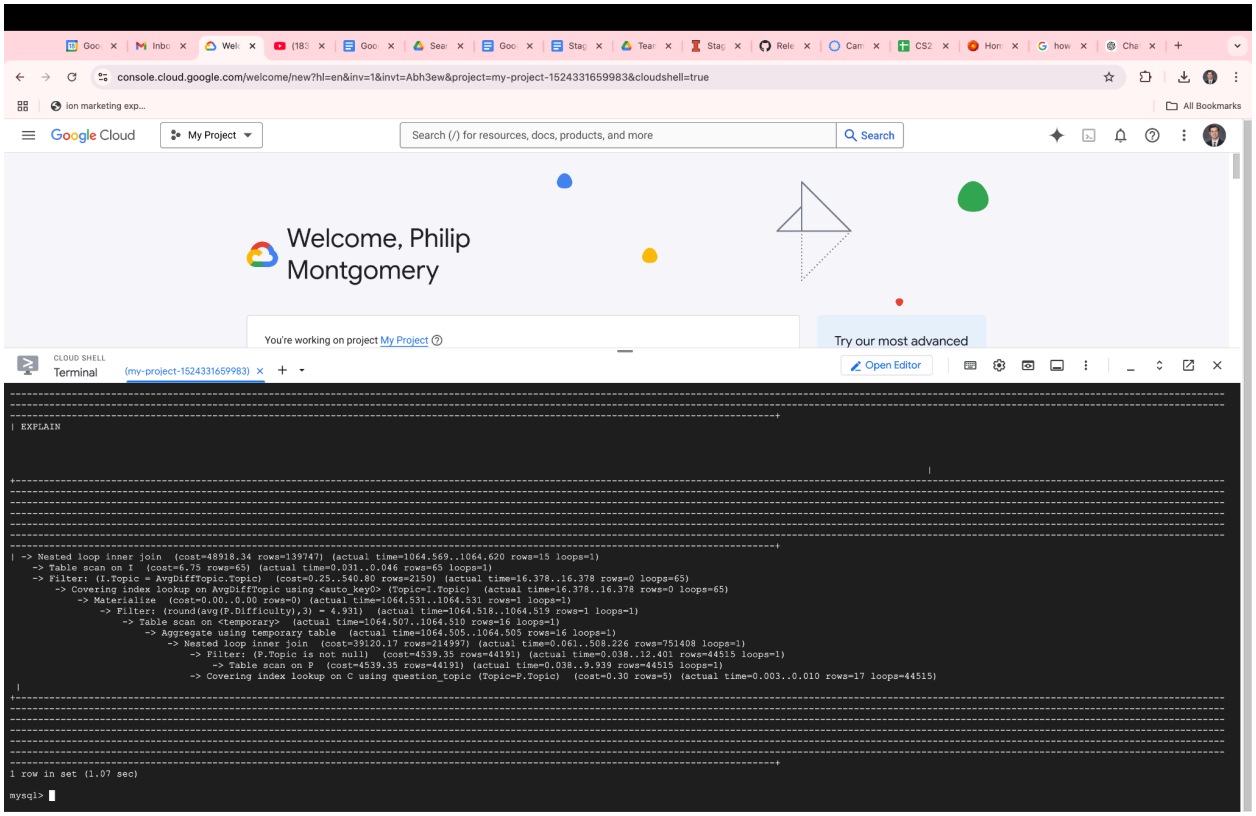
Final Index:

CREATE INDEX question_topic ON Courses(Topic)



Indexing for Query:

Select I.WebURL FROM Info I natural join (SELECT C.Topic FROM Courses C natural join Problems P GROUP BY C.Topic HAVING ROUND(AVG(P.Difficulty),3) = 4.931) AS AvgDiffTopic

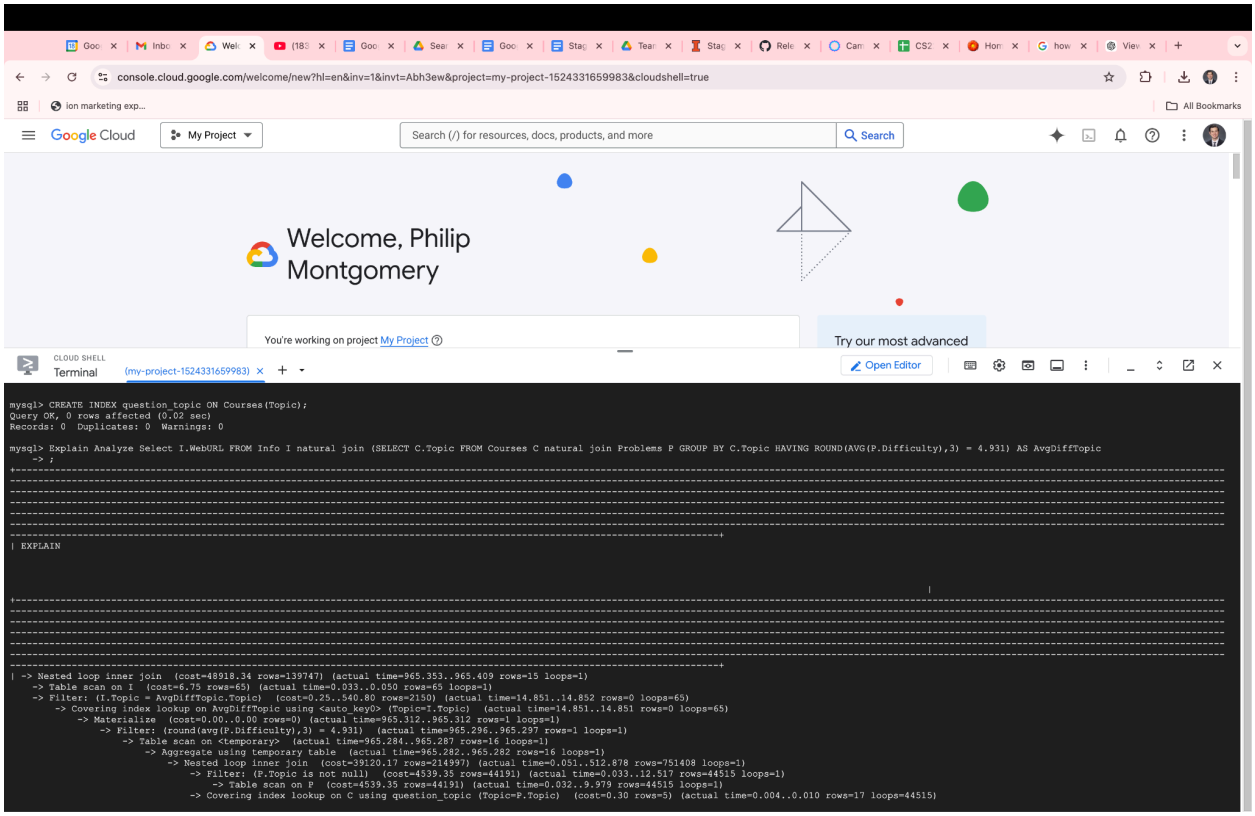


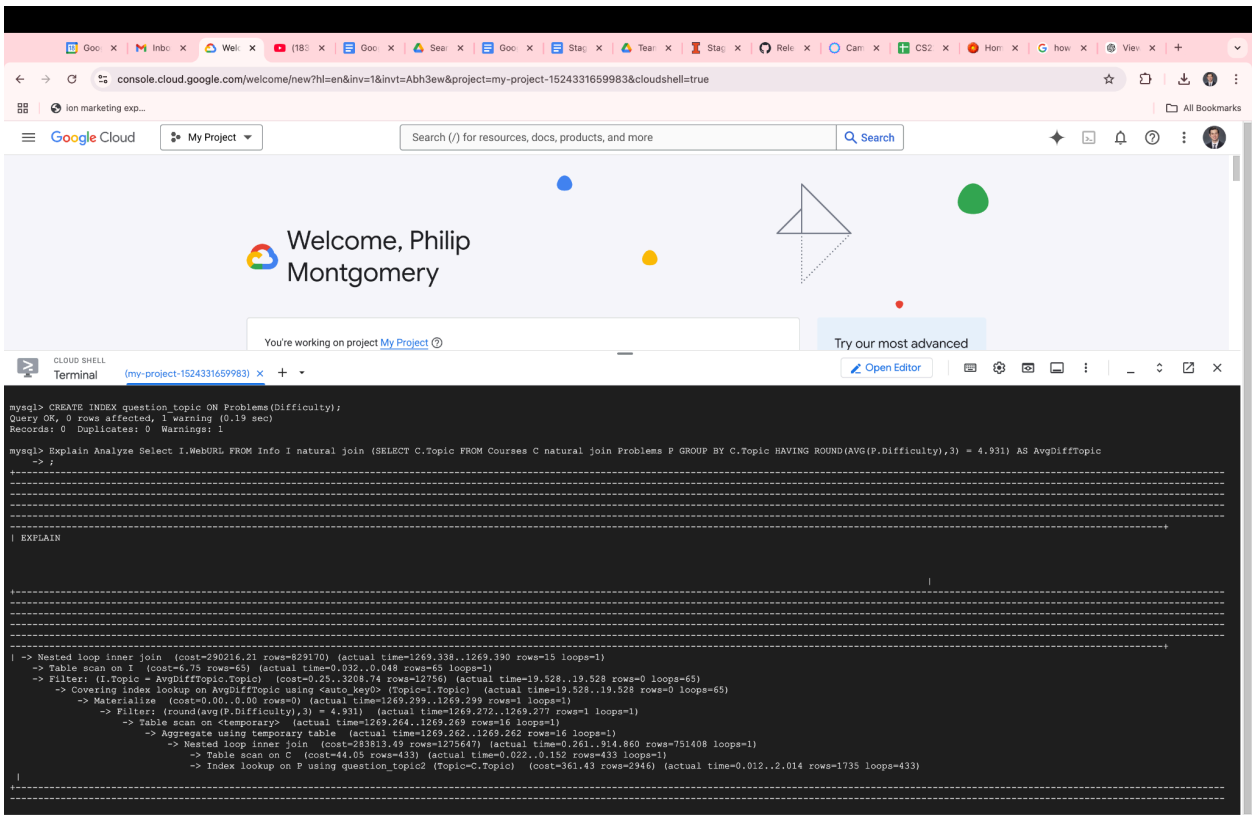
Indexes Tried:

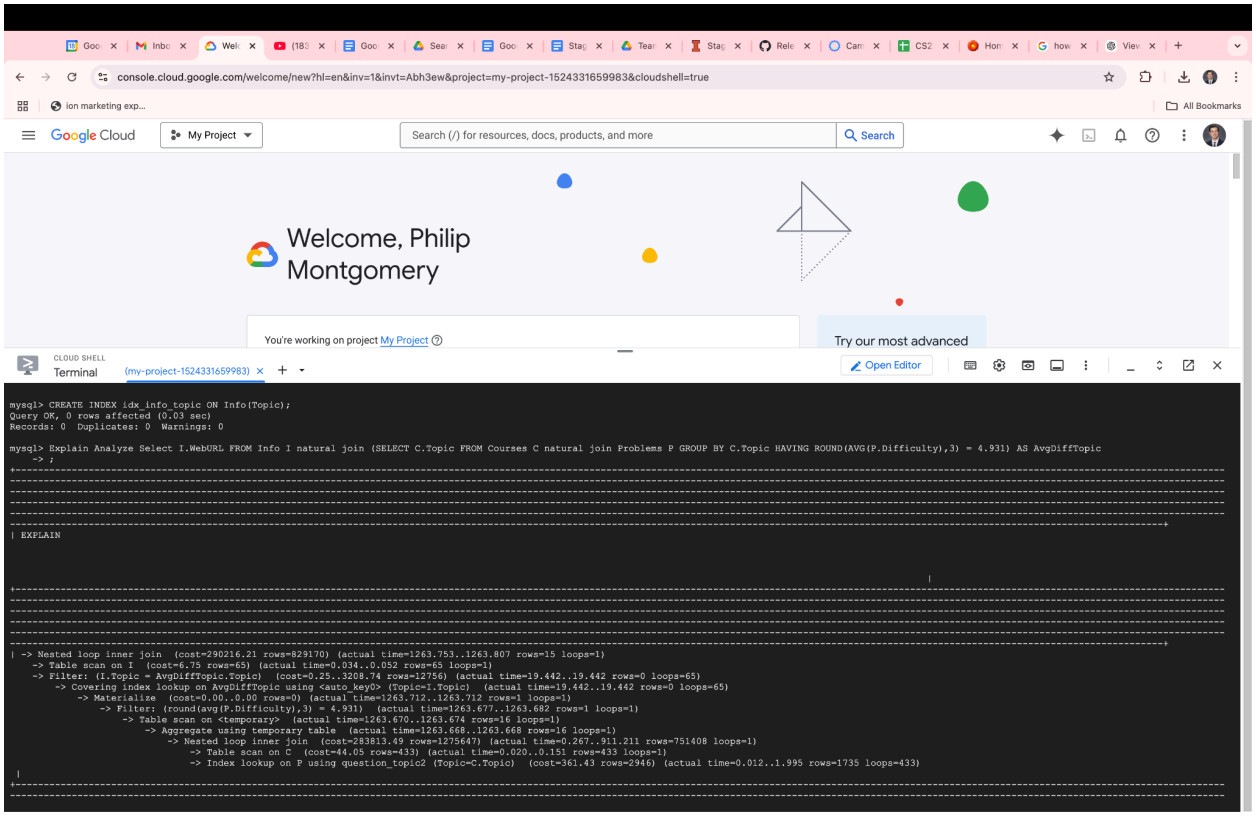
Better : CREATE INDEX question_topic ON Courses(Topic);

Worse : CREATE INDEX question_topic ON Problems(Difficulty);

No Effect: CREATE INDEX idx_info_topic ON Info(Topic);


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
Welcome, Philip Montgomery
You're working on project My Project
Try our most advanced
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
mysql> CREATE INDEX question_topic ON Courses(Topic);
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> Explain Analyze Select I.WebURL FROM Info I natural join (SELECT C.Topic FROM Courses C natural join Problems P GROUP BY C.Topic HAVING ROUND(AVG(P.Difficulty),3) = 4.931) AS AvgDiffTopic
-> ;
| EXPLAIN
| -> Nested loop inner join (cost=48918.34 rows=139747) (actual time=965.353..965.409 rows=15 loops=1)
-> Table scan on I (cost=6.75 rows=65) (actual time=0.033..0.050 rows=65 loops=1)
-> Filter: (I.Topic = AvgDiffTopic.Topic) (cost=0.25..540.80 rows=2150) (actual time=14.851..14.852 rows=0 loops=65)
-> Covering index lookup on AvgDiffTopic using <auto_key0> (Topic=I.Topic) (actual time=14.851..14.851 rows=0 loops=65)
-> Materialize (cost=0.00..0.00 rows=0) (actual time=965.312..965.312 rows=1 loops=1)
-> Filter: (round(avg(P.Difficulty),3) = 4.931) (actual time=965.296..965.297 rows=1 loops=1)
-> Table scan on <temporary> (actual time=965.284..965.287 rows=16 loops=1)
-> Aggregate using temporary table (actual time=965.282..965.282 rows=16 loops=1)
-> Nested loop inner join (cost=39120.17 rows=214997) (actual time=0.051..512.878 rows=751408 loops=1)
-> Filter: (P.Topic is not null) (cost=4539.35 rows=44191) (actual time=0.033..12.517 rows=44515 loops=1)
-> Table scan on P (cost=4539.35 rows=44191) (actual time=0.032..9.979 rows=44515 loops=1)
-> Covering index lookup on C using question_topic (Topic=P.Topic) (cost=0.30 rows=5) (actual time=0.004..0.010 rows=17 loops=44515)


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
Welcome, Philip Montgomery
You're working on project My Project
Try our most advanced
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
mysql> CREATE INDEX question_topic ON Problems(Difficulty);
Query OK, 0 rows affected, 1 warning (0.19 sec)
Records: 0 Duplicates: 0 Warnings: 1
mysql> Explain Analyze Select I.WebURL FROM Info I natural join (SELECT C.Topic FROM Courses C natural join Problems P GROUP BY C.Topic HAVING ROUND(AVG(P.Difficulty),3) = 4.931) AS AvgDiffTopic
-> ;
| EXPLAIN
| -> Nested loop inner join (cost=290216.21 rows=829170) (actual time=1269.338..1269.390 rows=15 loops=1)
-> Table scan on I (cost=6.75 rows=65) (actual time=0.032..0.048 rows=65 loops=1)
-> Filter: (I.Topic = AvgDiffTopic.Topic) (cost=0.25..3208.74 rows=12756) (actual time=19.528..19.528 rows=0 loops=65)
-> Covering index lookup on AvgDiffTopic using <auto_key0> (Topic=I.Topic) (actual time=19.528..19.528 rows=0 loops=65)
-> Materialize (cost=0.00..0.00 rows=0) (actual time=1269.299..1269.299 rows=1 loops=1)
-> Filter: (round(avg(P.Difficulty),3) = 4.931) (actual time=1269.272..1269.277 rows=1 loops=1)
-> Table scan on <temporary> (actual time=1269.264..1269.269 rows=16 loops=1)
-> Aggregate using temporary table (actual time=1269.262..1269.262 rows=16 loops=1)
-> Nested loop inner join (cost=283813.49 rows=1275647) (actual time=0.261..914.860 rows=751408 loops=1)
-> Table scan on C (cost=44.05 rows=433) (actual time=0.022..0.152 rows=433 loops=1)
-> Index lookup on P using question_topic2 (Topic=C.Topic) (cost=361.43 rows=2946) (actual time=0.012..2.014 rows=1735 loops=433)



Explanation:

I tried 3 different indices in order to narrow down the optimal index. I tried three indices which I believed to be promising: CREATE INDEX question_topic ON Courses(Topic), CREATE INDEX question_topic ON Problems(Difficulty), and CREATE INDEX idx_info_topic ON Info(Topic).

First, I attempted to index on Courses(Topic), because attempts on previous queries where Courses(Topic) is naturally joined have led to lower cost. A similar effect was observed here where the overall cost was reduced: The original cost was cost=290216.21 and the reduced time cost was cost=48918.34. This is likely due to the benefits from faster lookup.

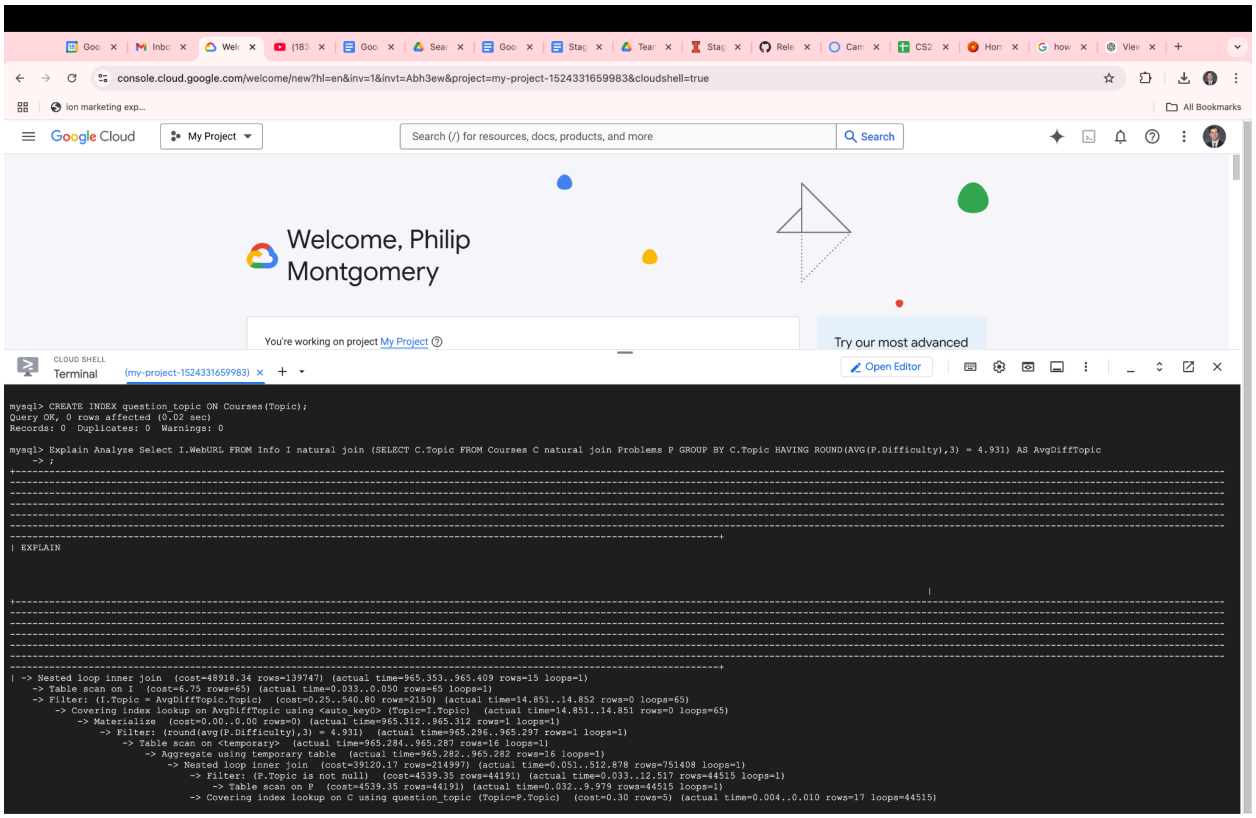
Second, I attempted to index on Problems(Difficulty). The logic behind this was to reduce the cost of performing the Round(Avg(P.Difficulty)). This turned out to not be the case though and actually increased the cost of performing the relevant operations going from a cost of 39120 to 283813. This is likely due to the fact that the row is aggregating when it takes the average so it needs to access all the data elements anyway, so indexing does not help, and in this case harms that process.

Finally, I attempted CREATE INDEX idx_info_topic ON Info(Topic). Which ended up having no effect. Double indexing on the same operation seems to have no real benefit for the actual performance. As it is only performing one comparison from the smaller table to the larger table, rather than doing two comparisons in both directions. This makes sense as it would be very cost inefficient to do some form of dual checking.

Overall the only index I used was CREATE INDEX question_topic ON Courses(Topic)

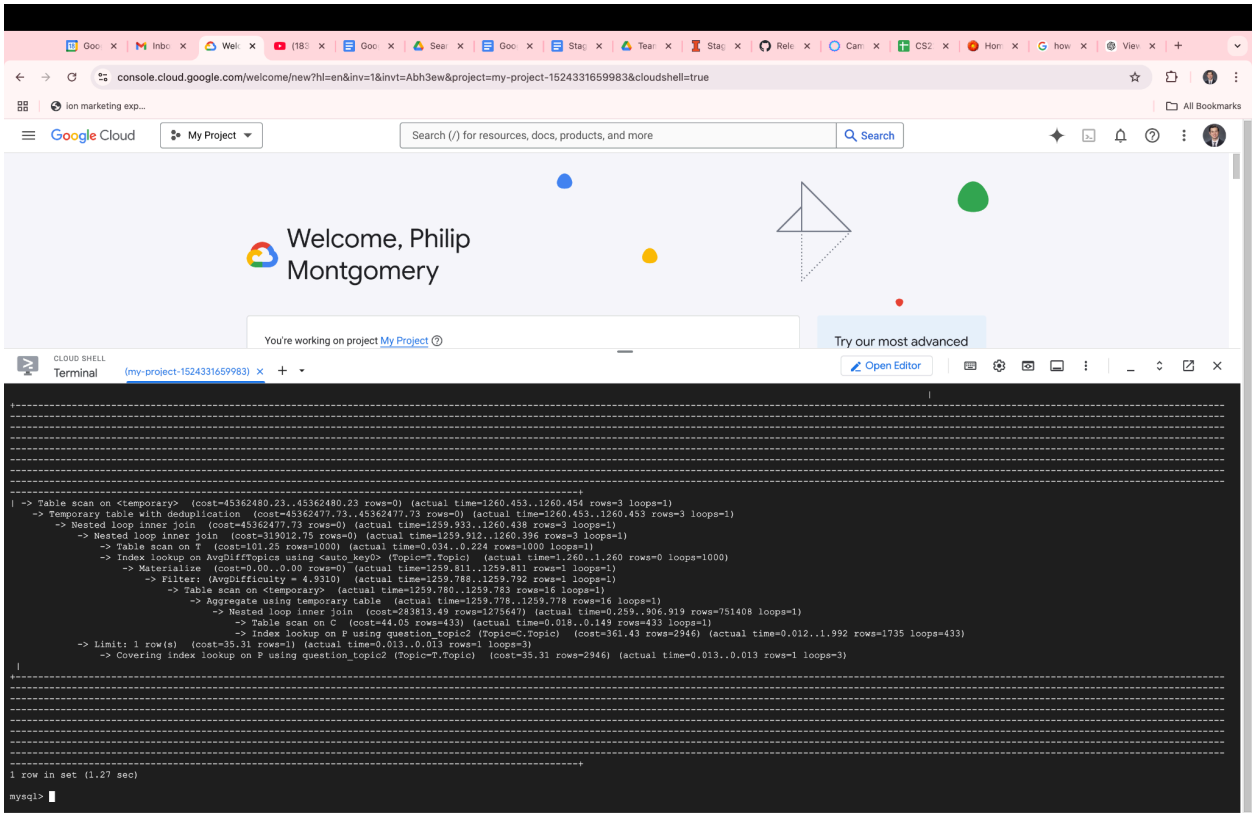
Final Index:

CREATE INDEX question_topic ON Courses(Topic);



Indexing for Query:

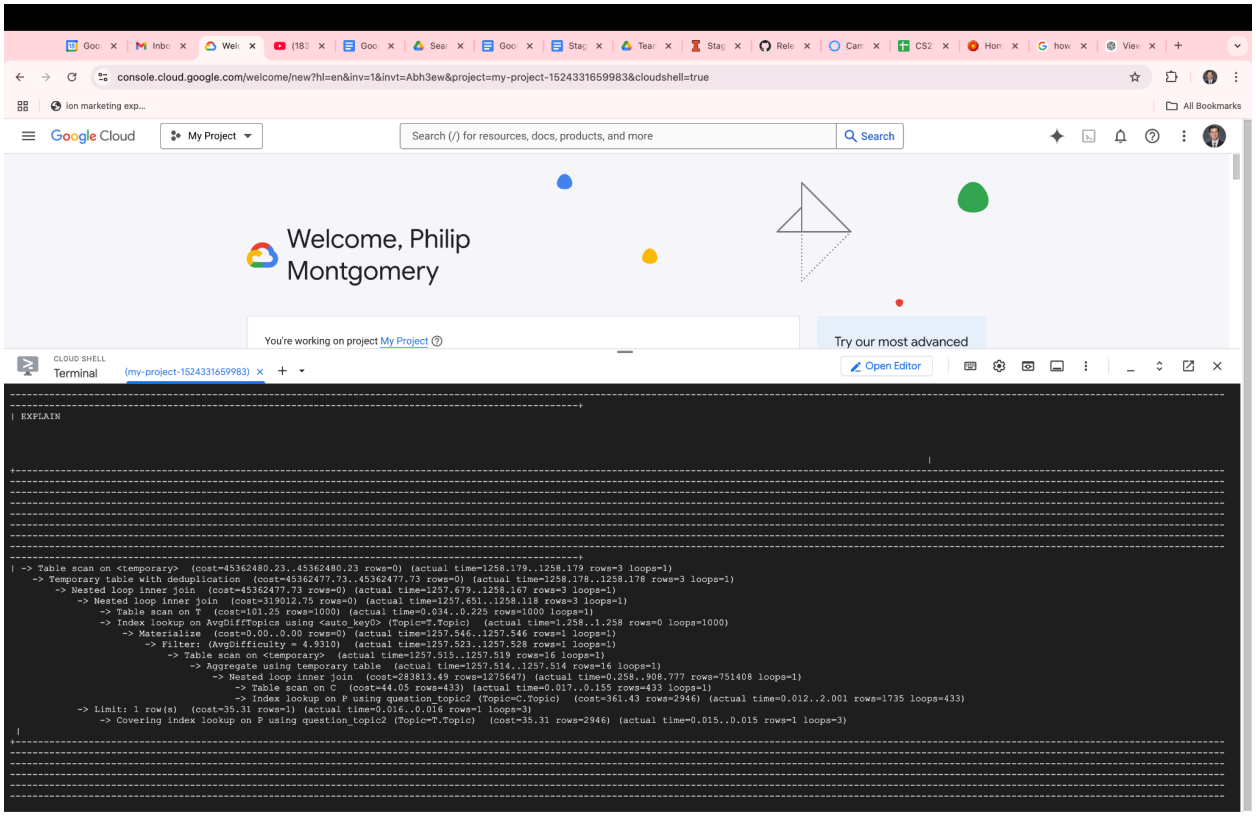
Select Distinct T.TutorId FROM Tutor T natural join Problems P JOIN(SELECT C.Topic, ROUND(AVG(P.Difficulty), 3) AS AvgDifficulty FROM Courses C JOIN Problems P ON C.Topic = P.Topic GROUP BY C.Topic HAVING AvgDifficulty = 4.9310) AS AvgDiffTopics ON P.Topic = AvgDiffTopics.Topic;

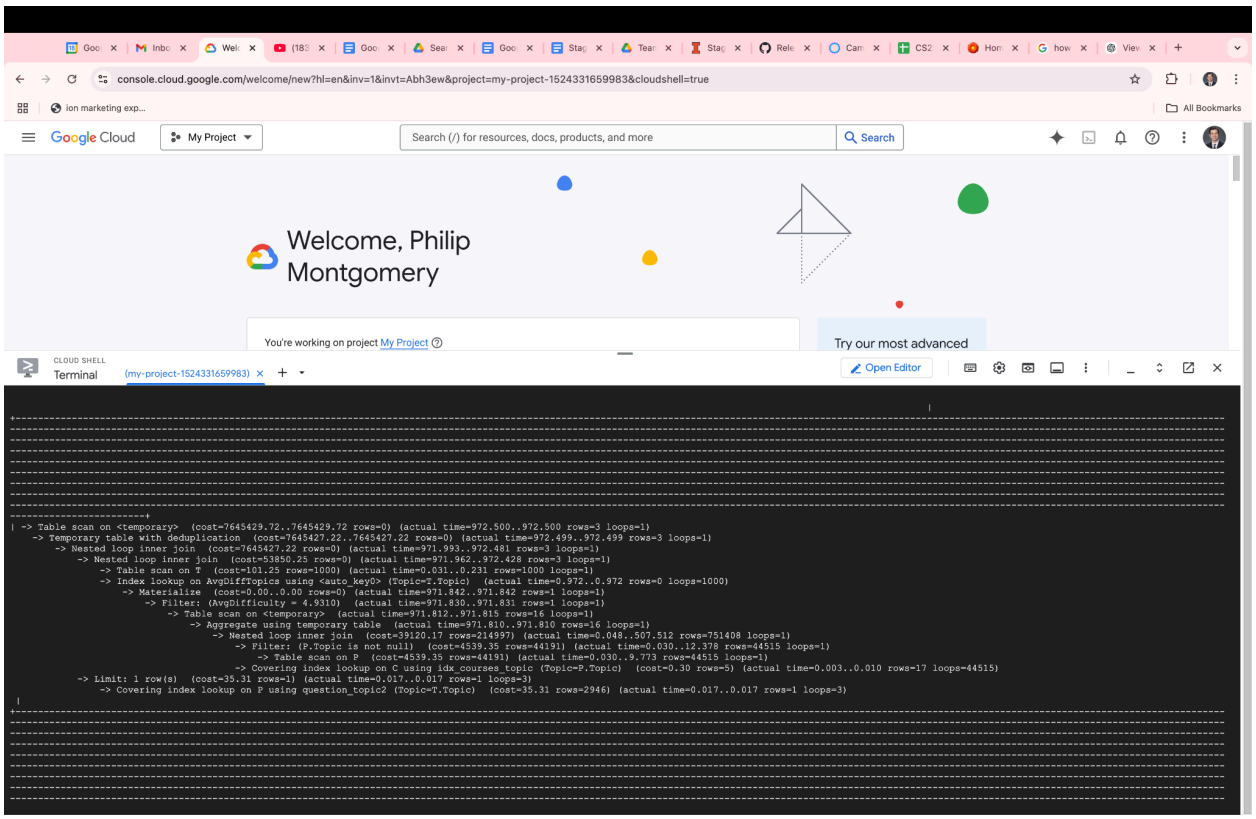


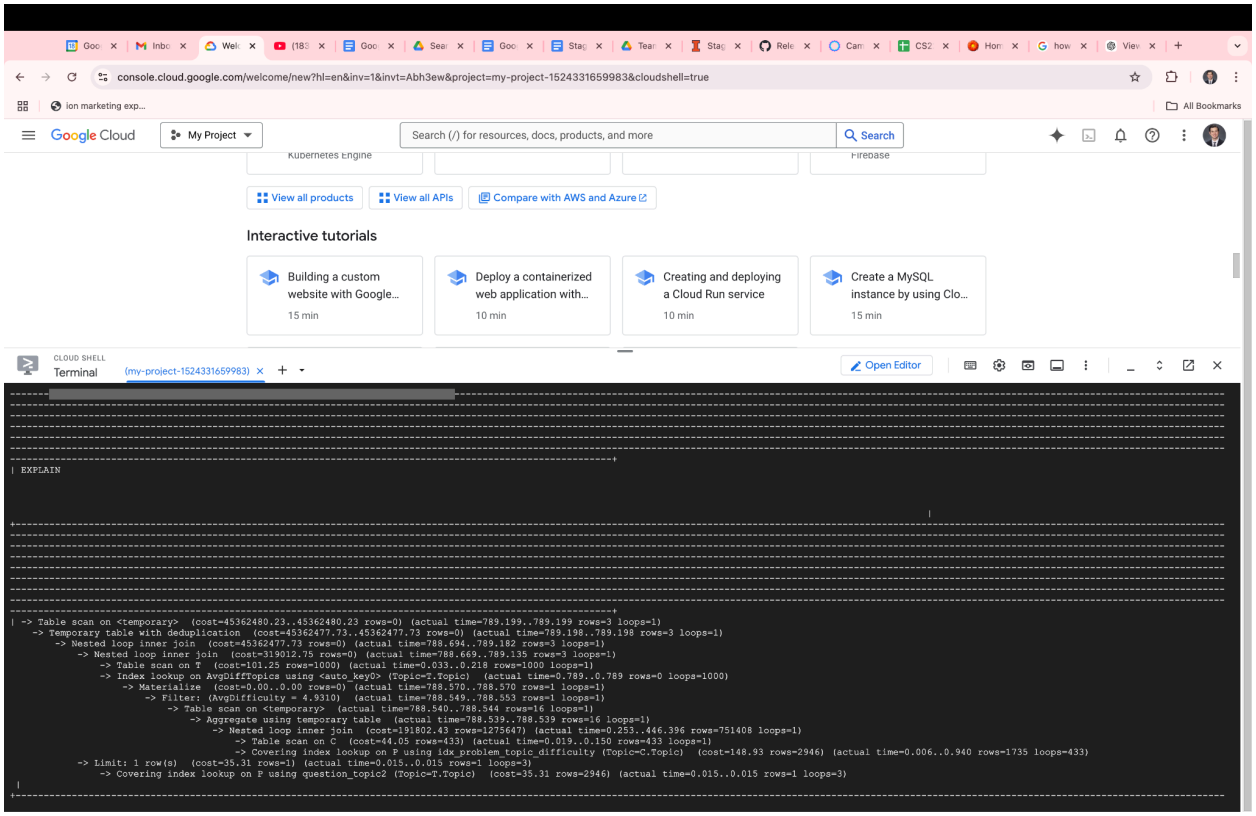
Same: CREATE INDEX idx_tutor_id ON Tutor(TutorId)

Better: CREATE INDEX idx_courses_topic ON Courses(Topic);

Slightly Better: CREATE INDEX idx_problem_topic_difficulty ON Problems(Topic, Difficulty)


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
Welcome, Philip Montgomery
You're working on project My Project
Try our most advanced
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
EXPLAIN
-> Table scan on <temporary> (cost=45362480.23..45362480.23 rows=0) (actual time=1258.179..1258.179 rows=3 loops=1)
-> Temporary table with deduplication (cost=45362477.73..45362477.73 rows=0) (actual time=1258.178..1258.178 rows=3 loops=1)
-> Nested loop inner join (cost=45362477.73 rows=0) (actual time=1257.679..1258.167 rows=3 loops=1)
-> Nested loop inner join (cost=319012.75 rows=0) (actual time=1257.651..1258.118 rows=3 loops=1)
-> Table scan on T (cost=101.25 rows=1000) (actual time=0.034..0.225 rows=1000 loops=1)
-> Index lookup on AvgDiffTopics using <auto_key0> (Topic=T.Topic) (actual time=1.258..1.258 rows=0 loops=1000)
-> Materialize (cost=0.00..0.00 rows=0) (actual time=1257.546..1257.546 rows=1 loops=1)
-> Filter: (AvgDifficulty = 4.9310) (actual time=1257.523..1257.528 rows=1 loops=1)
-> Table scan on <temporary> (actual time=1257.515..1257.519 rows=16 loops=1)
-> Aggregate using temporary table (actual time=1257.514..1257.514 rows=16 loops=1)
-> Nested loop inner join (cost=283813.49 rows=1275647) (actual time=0.258..908.777 rows=751408 loops=1)
-> Table scan on C (cost=44.05 rows=433) (actual time=0.017..0.155 rows=433 loops=1)
-> Index lookup on P using question_topic2 (Topic=C.Topic) (cost=361.43 rows=2946) (actual time=0.012..2.001 rows=1735 loops=433)
-> Limit: 1 row(s) (cost=35.31 rows=1) (actual time=0.016..0.016 rows=1 loops=3)
-> Covering index lookup on P using question_topic2 (Topic=T.Topic) (cost=35.31 rows=2946) (actual time=0.015..0.015 rows=1 loops=3)


console.cloud.google.com/welcome/new?hl=en&inv=1&invt=Abh3ew&project=my-project-1524331659983&cloudshell=true
Google Cloud
My Project
Search (/) for resources, docs, products, and more
Search
Welcome, Philip Montgomery
You're working on project My Project
Try our most advanced
CLOUD SHELL
Terminal
(my-project-1524331659983)
Open Editor
-> Table scan on <temporary> (cost=7645429.72..7645429.72 rows=0) (actual time=972.500..972.500 rows=3 loops=1)
-> Temporary table with deduplication (cost=7645427.22..7645427.22 rows=0) (actual time=972.499..972.499 rows=3 loops=1)
-> Nested loop inner join (cost=7645427.22 rows=0) (actual time=971.993..972.481 rows=3 loops=1)
-> Nested loop inner join (cost=53850.25 rows=0) (actual time=971.962..972.428 rows=3 loops=1)
-> Table scan on T (cost=101.25 rows=1000) (actual time=0.031..0.231 rows=1000 loops=1)
-> Index lookup on AvgDiffTopics using <auto_key0> (Topic=T.Topic) (actual time=0.972..0.972 rows=0 loops=1000)
-> Materialize (cost=0.00..0.00 rows=0) (actual time=971.842..971.842 rows=1 loops=1)
-> Filter: (AvgDifficulty = 4.9310) (actual time=971.830..971.831 rows=1 loops=1)
-> Table scan on <temporary> (actual time=971.812..971.815 rows=16 loops=1)
-> Aggregate using temporary table (actual time=971.810..971.810 rows=16 loops=1)
-> Nested loop inner join (cost=39120.17 rows=214997) (actual time=0.048..507.512 rows=751408 loops=1)
-> Filter: (P.Topic is not null) (cost=4539.35 rows=44191) (actual time=0.030..12.378 rows=44515 loops=1)
-> Table scan on P (cost=4539.35 rows=44191) (actual time=0.030..9.773 rows=44515 loops=1)
-> Covering index lookup on C using idx_courses_topic (Topic=P.Topic) (cost=0.30 rows=5) (actual time=0.003..0.010 rows=17 loops=44515)
-> Limit: 1 row(s) (cost=35.31 rows=1) (actual time=0.017..0.017 rows=1 loops=3)
-> Covering index lookup on P using question_topic2 (Topic=T.Topic) (cost=35.31 rows=2946) (actual time=0.017..0.017 rows=1 loops=3)



Explanation:

I attempted 3 different indices:CREATE INDEX idx_tutor_id ON Tutor(TutorId), CREATE INDEX idx_courses_topic ON Courses(Topic), CREATE INDEX idx_problem_topic_difficulty ON Problems(Topic, Difficulty).

First, I attempted to index on Tutor(TutorId) , but there was no noticeable change in the results of the cost. This is likely due to the fact that primary keys are already indexed.

Next, I attempted to do Courses(Topic), which similar to the other indices led to a massive decrease in the cost. The cost decreased from cost=45362480.23 to cost=7645429.72. This massive decrease is due to similar reasons as previous queries: there were far more problems being pulled than in the previous query thus magnifying the impact of the cost saving “look up table” indexing.

Finally I tried to perform a composite index on Problems(Topic, Difficulty) since we are Grouping by the avg difficulty and topic. There seemed to be a small decrease in cost due to performing this index: the next inner loop join cost went down from a cost of 283813 to 191802. This is likely due to the more efficient range scan to get both Topic and difficulty when the inner join occurs.

Overall I ended up using both CREATE INDEX idx_courses_topic ON Courses(Topic) and CREATE INDEX idx_problem_topic_difficulty ON Problems(Topic, Difficulty).

Final Index:

CREATE INDEX idx_courses_topic ON Courses(Topic);
CREATE INDEX idx_problem_topic_difficulty ON Problems(Topic, Difficulty)