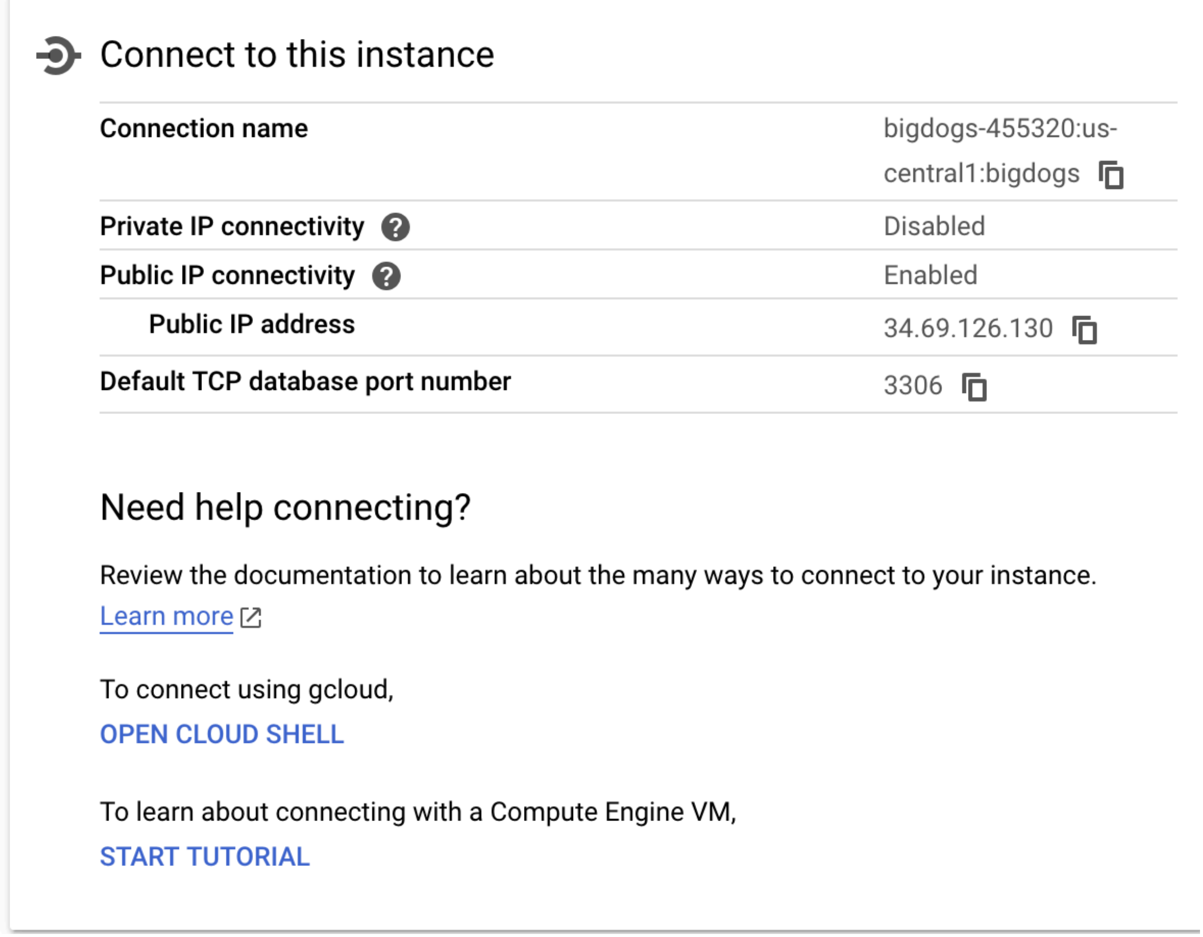
Part 1

```
CREATE TABLE User_Info (
        Email VARCHAR(255) PRIMARY KEY,
        Name VARCHAR(255),
        Password VARCHAR(255)
);

CREATE TABLE User_Chats (
        ChatID VARCHAR(255),
        Email VARCHAR(255),
        Password VARCHAR(255),
        FOREIGN KEY (Email) REFERENCES User_Info(Email)
);

CREATE TABLE Patient_Diagnosis_Records (
        RecordID VARCHAR(255) PRIMARY KEY,
        VisitDescription VARCHAR(1000),
        DoctorSpecialty VARCHAR(255),
        MedicalTranscription VARCHAR(1000),
        Keywords VARCHAR(750)
);

CREATE TABLE Question_Answer_Symptoms (
        QuestionID VARCHAR(255) PRIMARY KEY,
        Question VARCHAR(1000),
        Answer VARCHAR(1000),
        FocusArea VARCHAR(255)
);

CREATE TABLE Search_Log (
        SearchID VARCHAR(255) PRIMARY KEY,
        Email VARCHAR(255),
        Search_text TEXT(10000),
        FOREIGN KEY (Email) REFERENCES User_Info(Email)
);

CREATE TABLE Log_to_Patient (
        SearchID VARCHAR(255),
        RecordID VARCHAR(255),
        PRIMARY KEY (SearchID, RecordID),
        FOREIGN KEY (SearchID) REFERENCES Search_Log(SearchID),
        FOREIGN KEY (RecordID) REFERENCES
Patient_Diagnosis_Records(RecordID)
);
```

Proof of Work



Connect to this instance

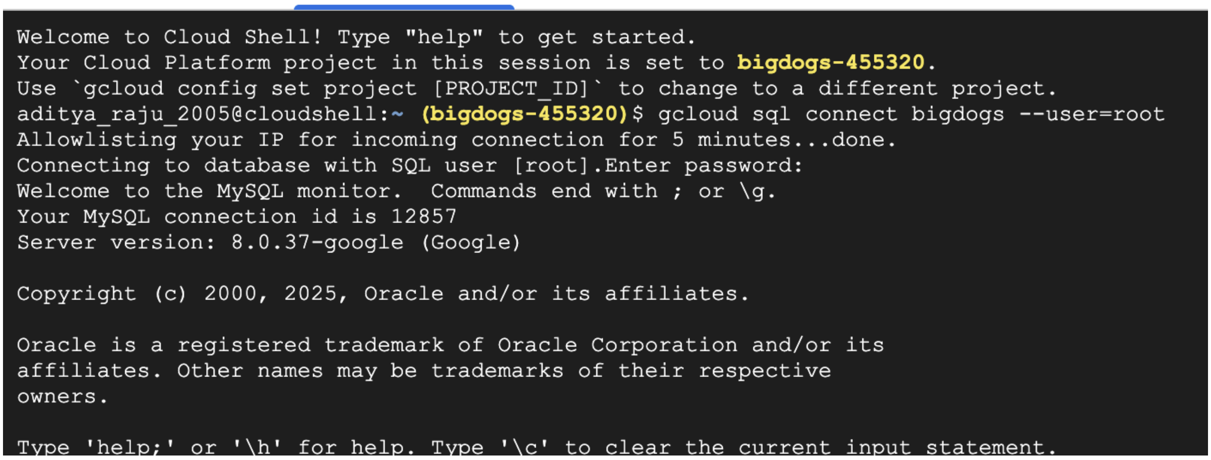
Connection name	bigdogs-455320:us-central1:bigdogs
Private IP connectivity	Disabled
Public IP connectivity	Enabled
Public IP address	34.69.126.130
Default TCP database port number	3306

Need help connecting?

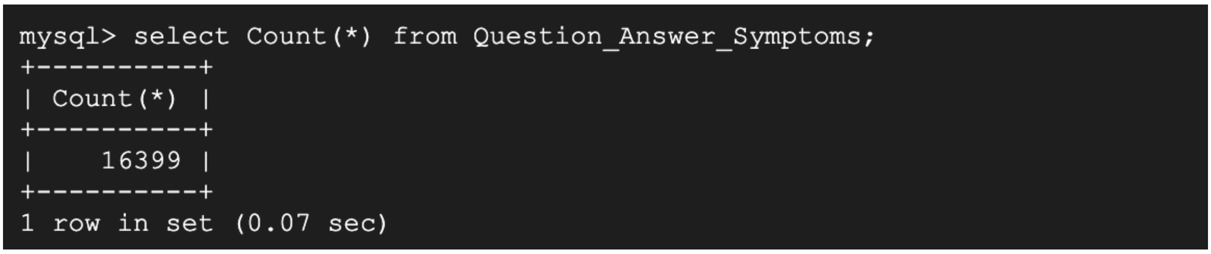
Review the documentation to learn about the many ways to connect to your instance.
Learn more

To connect using gcloud,
OPEN CLOUD SHELL

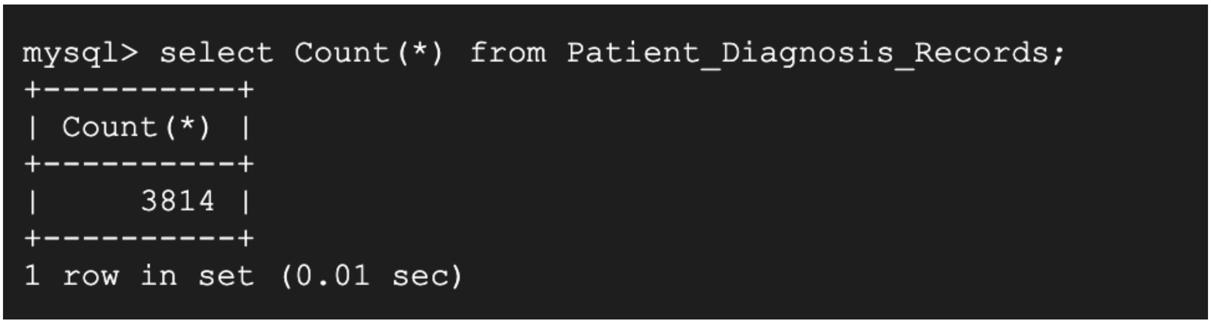
To learn about connecting with a Compute Engine VM,
START TUTORIAL



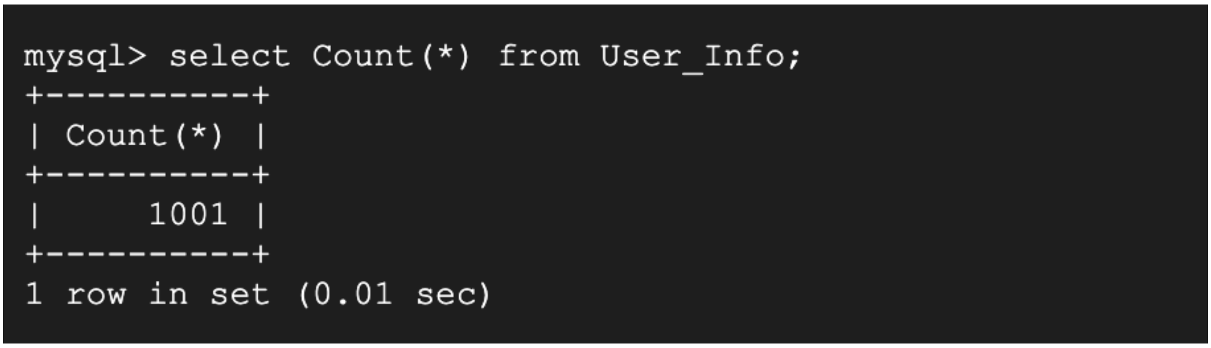
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to bigdogs-455320.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
aditya_raju_2005@cloudshell:~ (bigdogs-455320)$ gcloud sql connect bigdogs --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12857
Server version: 8.0.37-google (Google)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```



```
mysql> select Count(*) from Question_Answer_Symptoms;
+----------+
| Count(*) |
+----------+
|    16399 |
+----------+
1 row in set (0.07 sec)
```



```
mysql> select Count(*) from Patient_Diagnosis_Records;
+----------+
| Count(*) |
+----------+
|     3814 |
+----------+
1 row in set (0.01 sec)
```



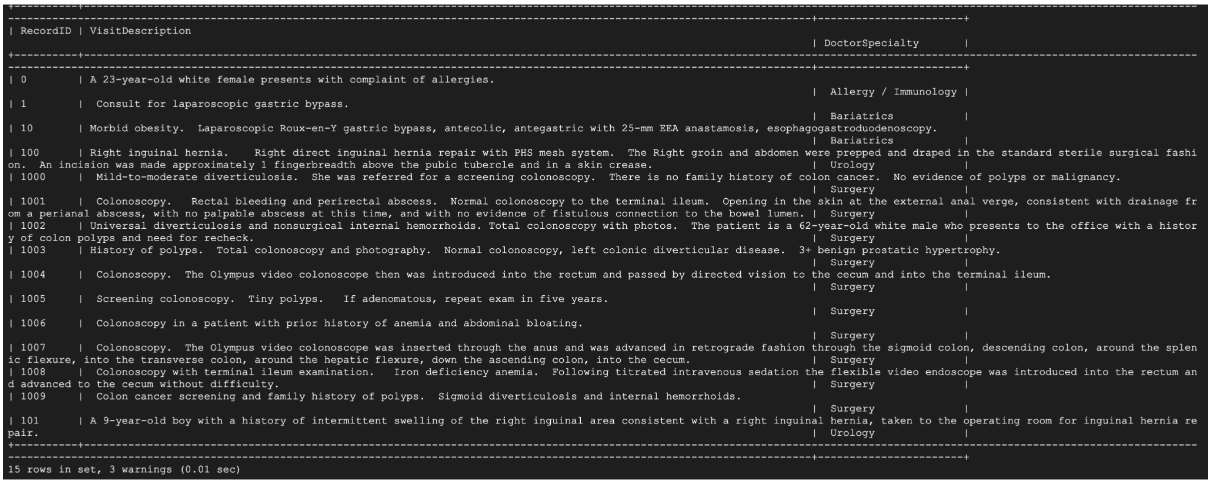
```
mysql> select Count(*) from User_Info;
+----------+
| Count(*) |
+----------+
|     1001 |
+----------+
1 row in set (0.01 sec)
```

We only have data in these 3 different tables so far. The others are empty as they will only populate as users use our application.

Advanced Queries:

Query 1: Find all Patient Diagnosis Records matching the Focus Area of a specific Question.

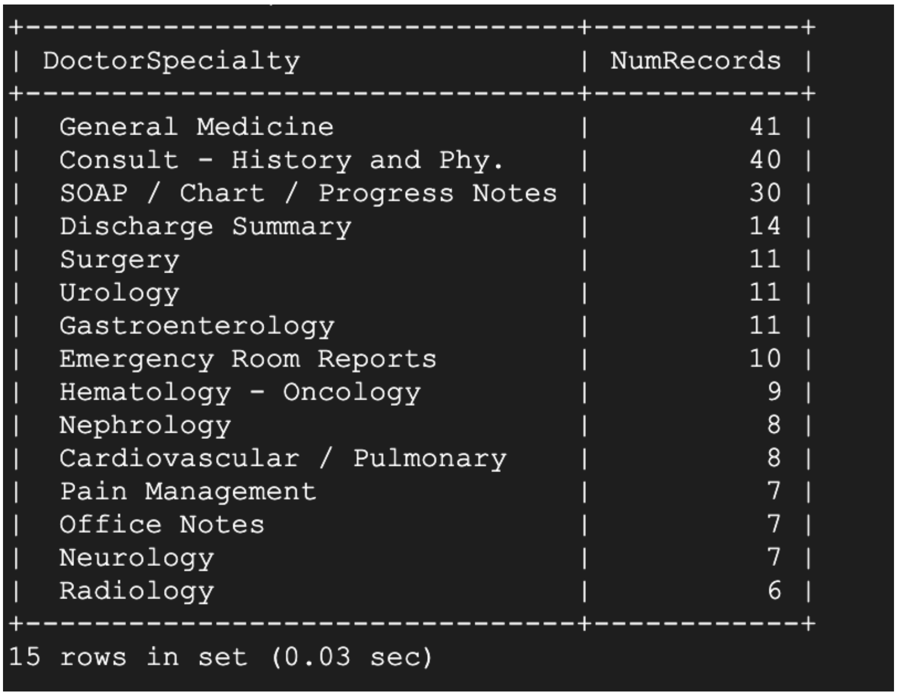
```
SELECT DISTINCT PDR.RecordID, PDR.VisitDescription, PDR.DoctorSpecialty
FROM Patient_Diagnosis_Records PDR
INNER JOIN Question_Answer_Symptoms QAS
  ON PDR.Keywords LIKE '%' || QAS.FocusArea || '%'
WHERE QAS.QuestionID = (
    SELECT QuestionID
    FROM Question_Answer_Symptoms
    WHERE Question LIKE '%headache%'
    LIMIT 1
);
```



```
+----------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------+-----------------------+
| RecordID | VisitDescription
                                                                                                                               | DoctorSpecialty       |
+----------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------+-----------------------+
| 0        | A 23-year-old white female presents with complaint of allergies.
                                                                                                                               |  Allergy / Immunology |
| 1        |  Consult for laparoscopic gastric bypass.
                                                                                                                               |  Bariatrics           |
| 10       | Morbid obesity.  Laparoscopic Roux-en-Y gastric bypass, antecolic, antegastric with 25-mm EEA anastamosis, esophagogastroduodenoscopy.
                                                                                                                               |  Bariatrics           |
| 100      | Right inguinal hernia.    Right direct inguinal hernia repair with PHS mesh system.  The Right groin and abdomen were prepped and draped in the standard sterile surgical fashi
on.  An incision was made approximately 1 fingerbreadth above the pubic tubercle and in a skin crease.                         |  Urology              |
| 1000     |  Mild-to-moderate diverticulosis.  She was referred for a screening colonoscopy.  There is no family history of colon cancer.  No evidence of polyps or malignancy.
                                                                                                                               |  Surgery              |
| 1001     |  Colonoscopy.   Rectal bleeding and perirectal abscess.  Normal colonoscopy to the terminal ileum.  Opening in the skin at the external anal verge, consistent with drainage fr
om a perianal abscess, with no palpable abscess at this time, and with no evidence of fistulous connection to the bowel lumen. |  Surgery              |
| 1002     | Universal diverticulosis and nonsurgical internal hemorrhoids. Total colonoscopy with photos.  The patient is a 62-year-old white male who presents to the office with a histor
y of colon polyps and need for recheck.                                                                                        |  Surgery              |
| 1003     | History of polyps.  Total colonoscopy and photography.  Normal colonoscopy, left colonic diverticular disease.  3+ benign prostatic hypertrophy.
                                                                                                                               |  Surgery              |
| 1004     |  Colonoscopy.  The Olympus video colonoscope then was introduced into the rectum and passed by directed vision to the cecum and into the terminal ileum.
                                                                                                                               |  Surgery              |
| 1005     |  Screening colonoscopy.  Tiny polyps.   If adenomatous, repeat exam in five years.
                                                                                                                               |  Surgery              |
| 1006     |  Colonoscopy in a patient with prior history of anemia and abdominal bloating.
                                                                                                                               |  Surgery              |
| 1007     |  Colonoscopy.  The Olympus video colonoscope was inserted through the anus and was advanced in retrograde fashion through the sigmoid colon, descending colon, around the splen
ic flexure, into the transverse colon, around the hepatic flexure, down the ascending colon, into the cecum.                   |  Surgery              |
| 1008     |  Colonoscopy with terminal ileum examination.   Iron deficiency anemia.  Following titrated intravenous sedation the flexible video endoscope was introduced into the rectum an
d advanced to the cecum without difficulty.                                                                                    |  Surgery              |
| 1009     |  Colon cancer screening and family history of polyps.  Sigmoid diverticulosis and internal hemorrhoids.
                                                                                                                               |  Surgery              |
| 101      | A 9-year-old boy with a history of intermittent swelling of the right inguinal area consistent with a right inguinal hernia, taken to the operating room for inguinal hernia re
pair.                                                                                                                          |  Urology              |
+----------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------+-----------------------+
15 rows in set, 3 warnings (0.01 sec)
```

Query 2: Find DoctorSpecialties that have at least two records matching a user's keyword search (e.g., "fever"), and order them by the number of matching records.

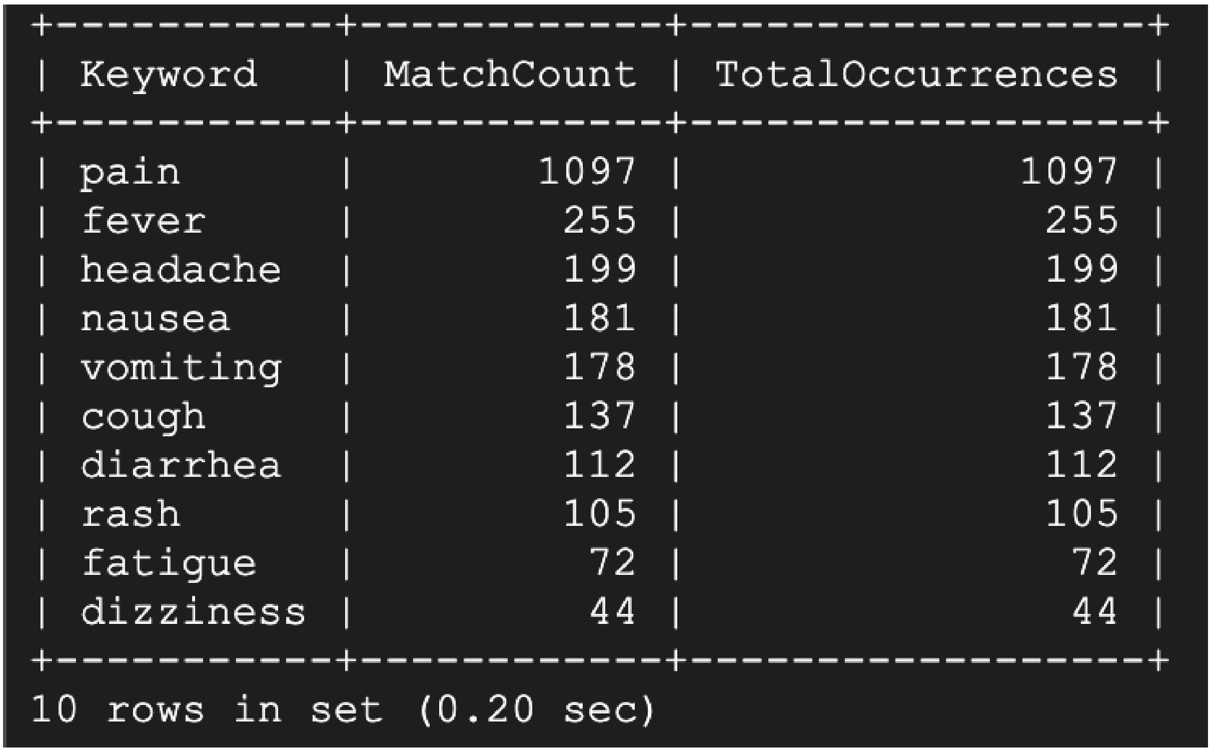
```
SELECT DoctorSpecialty, COUNT(*) AS NumRecords
FROM Patient_Diagnosis_Records
WHERE RecordID IN (
    SELECT RecordID
    FROM Patient_Diagnosis_Records
    WHERE MedicalTranscription LIKE '%fever%' OR Keywords LIKE '%fever%'
)
GROUP BY DoctorSpecialty
HAVING COUNT(*) >= 2
ORDER BY NumRecords DESC;
```



```
+--------------------------------+------------+
| DoctorSpecialty                | NumRecords |
+--------------------------------+------------+
|  General Medicine              |         41 |
|  Consult - History and Phy.    |         40 |
|  SOAP / Chart / Progress Notes |         30 |
|  Discharge Summary             |         14 |
|  Surgery                       |         11 |
|  Urology                       |         11 |
|  Gastroenterology              |         11 |
|  Emergency Room Reports        |         10 |
|  Hematology - Oncology         |          9 |
|  Nephrology                    |          8 |
|  Cardiovascular / Pulmonary    |          8 |
|  Pain Management               |          7 |
|  Office Notes                  |          7 |
|  Neurology                     |          7 |
|  Radiology                     |          6 |
+--------------------------------+------------+
15 rows in set (0.03 sec)
```

Query 3: Frequency of Top 10 Selected Medical Keywords Across Unique Records

```
SELECT k.Keyword, COUNT(DISTINCT pdr.RecordID) AS MatchCount, COUNT(*)
AS TotalOccurrences
FROM
   (
      SELECT 'fever' AS Keyword UNION ALL
      SELECT 'cough' UNION ALL
      SELECT 'headache' UNION ALL
      SELECT 'nausea' UNION ALL
      SELECT 'fatigue' UNION ALL
      SELECT 'pain' UNION ALL
      SELECT 'dizziness' UNION ALL
      SELECT 'rash' UNION ALL
      SELECT 'vomiting' UNION ALL
      SELECT 'diarrhea'
   ) AS k
JOIN Patient_Diagnosis_Records pdr ON pdr.Keywords LIKE CONCAT('%',
k.Keyword, '%')
GROUP BY k.Keyword
ORDER BY MatchCount DESC;
```

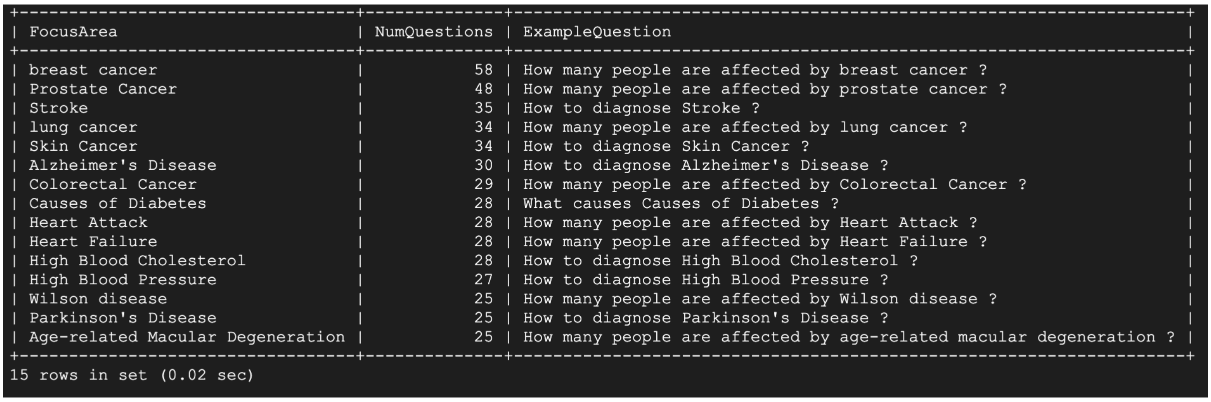


```
+-----------+------------+-------------------+
| Keyword   | MatchCount | TotalOccurrences  |
+-----------+------------+-------------------+
| pain      |       1097 |              1097 |
| fever     |        255 |               255 |
| headache  |        199 |               199 |
| nausea    |        181 |               181 |
| vomiting  |        178 |               178 |
| cough     |        137 |               137 |
| diarrhea  |        112 |               112 |
| rash      |        105 |               105 |
| fatigue   |         72 |                72 |
| dizziness |         44 |                44 |
+-----------+------------+-------------------+
10 rows in set (0.20 sec)
```

This query has only 10 rows of output.

Query 4: Most Common Question Texts and Their Associated FocusArea

```
SELECT
  TRIM(FocusArea) AS FocusArea,
  COUNT(*) AS NumQuestions,
  MIN(TRIM(Question)) AS ExampleQuestion
FROM Question_Answer_Symptoms
WHERE TRIM(FocusArea) IS NOT NULL AND TRIM(FocusArea) != ''
GROUP BY TRIM(FocusArea)
HAVING COUNT(*) > 1
ORDER BY NumQuestions DESC;
```

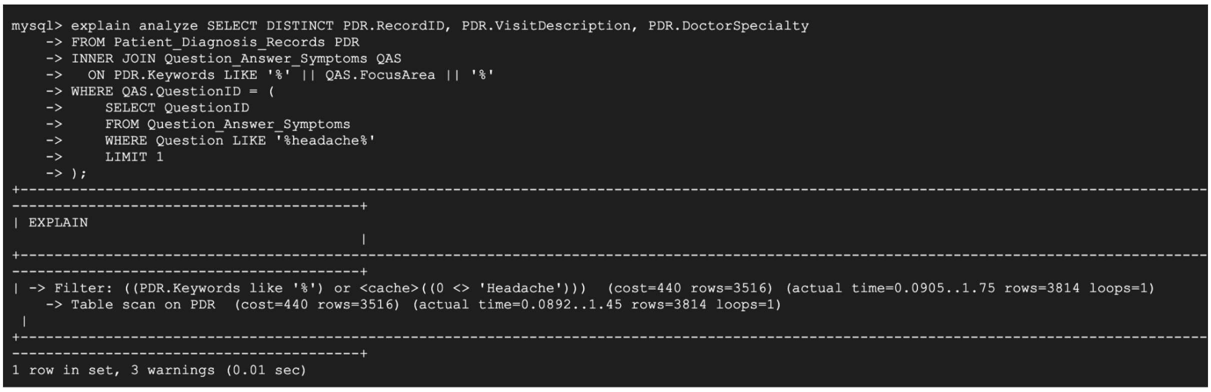


```
+----------------------------------+--------------+--------------------------------------------------------------------+
| FocusArea                        | NumQuestions | ExampleQuestion                                                    |
+----------------------------------+--------------+--------------------------------------------------------------------+
| breast cancer                    |           58 | How many people are affected by breast cancer ?                    |
| Prostate Cancer                  |           48 | How many people are affected by prostate cancer ?                  |
| Stroke                           |           35 | How to diagnose Stroke ?                                           |
| lung cancer                      |           34 | How many people are affected by lung cancer ?                      |
| Skin Cancer                      |           34 | How to diagnose Skin Cancer ?                                      |
| Alzheimer's Disease              |           30 | How to diagnose Alzheimer's Disease ?                              |
| Colorectal Cancer                |           29 | How many people are affected by Colorectal Cancer ?                |
| Causes of Diabetes               |           28 | What causes Causes of Diabetes ?                                   |
| Heart Attack                     |           28 | How many people are affected by Heart Attack ?                     |
| Heart Failure                    |           28 | How many people are affected by Heart Failure ?                    |
| High Blood Cholesterol           |           28 | How to diagnose High Blood Cholesterol ?                           |
| High Blood Pressure              |           27 | How to diagnose High Blood Pressure ?                              |
| Wilson disease                   |           25 | How many people are affected by Wilson disease ?                   |
| Parkinson's Disease              |           25 | How to diagnose Parkinson's Disease ?                              |
| Age-related Macular Degeneration |           25 | How many people are affected by age-related macular degeneration ? |
+----------------------------------+--------------+--------------------------------------------------------------------+
15 rows in set (0.02 sec)
```

Part 2

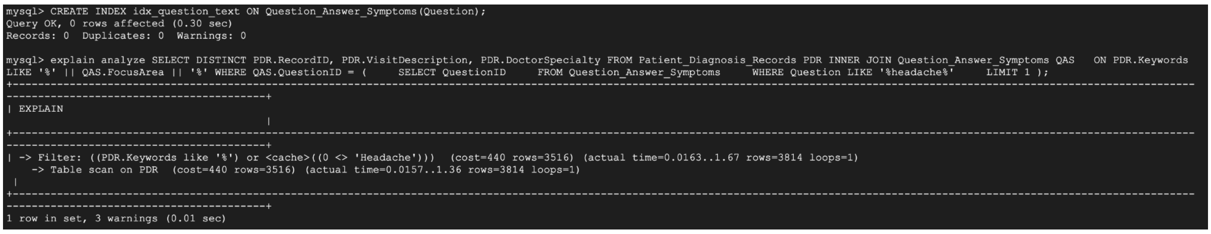
1) Indexing Query 1:

Performance without indexing:



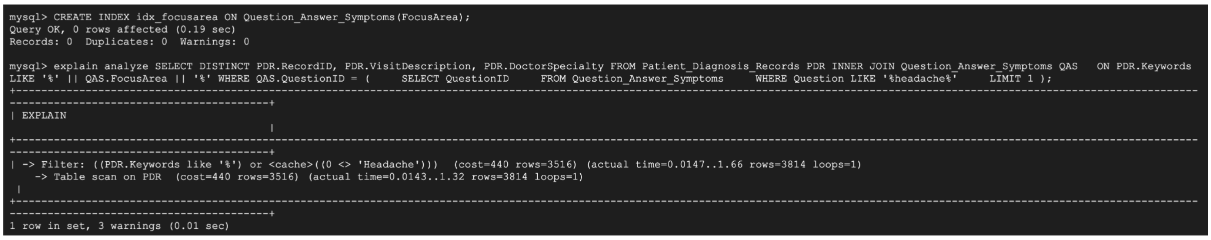
```
mysql> explain analyze SELECT DISTINCT PDR.RecordID, PDR.VisitDescription, PDR.DoctorSpecialty
    -> FROM Patient_Diagnosis_Records PDR
    -> INNER JOIN Question_Answer_Symptoms QAS
    ->   ON PDR.Keywords LIKE '%' || QAS.FocusArea || '%'
    -> WHERE QAS.QuestionID = (
    ->     SELECT QuestionID
    ->     FROM Question_Answer_Symptoms
    ->     WHERE Question LIKE '%headache%'
    ->     LIMIT 1
    -> );
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| EXPLAIN
                                         |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| -> Filter: ((PDR.Keywords like '%') or <cache>((0 <> 'Headache')))  (cost=440 rows=3516) (actual time=0.0905..1.75 rows=3814 loops=1)
    -> Table scan on PDR  (cost=440 rows=3516) (actual time=0.0892..1.45 rows=3814 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
1 row in set, 3 warnings (0.01 sec)
```

Performance after index on Question_Answer_Symptoms(Question):



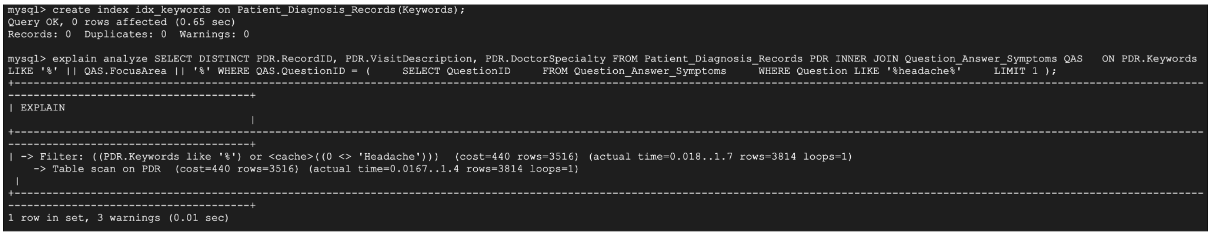
```
mysql> CREATE INDEX idx_question_text ON Question_Answer_Symptoms(Question);
Query OK, 0 rows affected (0.30 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT DISTINCT PDR.RecordID, PDR.VisitDescription, PDR.DoctorSpecialty FROM Patient_Diagnosis_Records PDR INNER JOIN Question_Answer_Symptoms QAS   ON PDR.Keywords
LIKE '%' || QAS.FocusArea || '%' WHERE QAS.QuestionID = (     SELECT QuestionID     FROM Question_Answer_Symptoms     WHERE Question LIKE '%headache%'     LIMIT 1 );
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| EXPLAIN
                                         |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| -> Filter: ((PDR.Keywords like '%') or <cache>((0 <> 'Headache')))  (cost=440 rows=3516) (actual time=0.0163..1.67 rows=3814 loops=1)
    -> Table scan on PDR  (cost=440 rows=3516) (actual time=0.0157..1.36 rows=3814 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
1 row in set, 3 warnings (0.01 sec)
```

Performance after index on Question_Answer_Symptoms(FocusArea)



```
mysql> CREATE INDEX idx_focusarea ON Question_Answer_Symptoms(FocusArea);
Query OK, 0 rows affected (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT DISTINCT PDR.RecordID, PDR.VisitDescription, PDR.DoctorSpecialty FROM Patient_Diagnosis_Records PDR INNER JOIN Question_Answer_Symptoms QAS   ON PDR.Keywords
LIKE '%' || QAS.FocusArea || '%' WHERE QAS.QuestionID = (     SELECT QuestionID     FROM Question_Answer_Symptoms     WHERE Question LIKE '%headache%'     LIMIT 1 );
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| EXPLAIN
                                         |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
| -> Filter: ((PDR.Keywords like '%') or <cache>((0 <> 'Headache')))  (cost=440 rows=3516) (actual time=0.0147..1.66 rows=3814 loops=1)
    -> Table scan on PDR  (cost=440 rows=3516) (actual time=0.0143..1.32 rows=3814 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------+
1 row in set, 3 warnings (0.01 sec)
```

Performance after index on Patient_Diagnosis_Records(Keywords)

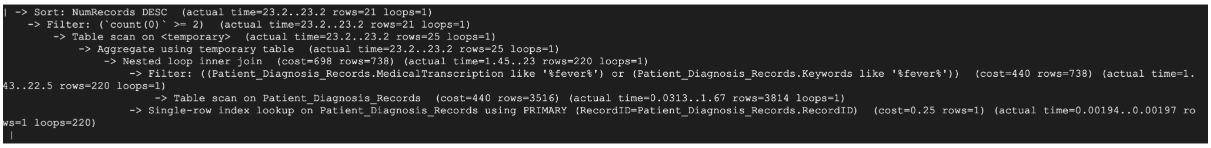


```
mysql> create index idx_keywords on Patient_Diagnosis_Records(Keywords);
Query OK, 0 rows affected (0.65 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain analyze SELECT DISTINCT PDR.RecordID, PDR.VisitDescription, PDR.DoctorSpecialty FROM Patient_Diagnosis_Records PDR INNER JOIN Question_Answer_Symptoms QAS   ON PDR.Keywords
LIKE '%' || QAS.FocusArea || '%' WHERE QAS.QuestionID = (     SELECT QuestionID     FROM Question_Answer_Symptoms     WHERE Question LIKE '%headache%'     LIMIT 1 );
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| EXPLAIN
                                     |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Filter: ((PDR.Keywords like '%') or <cache>((0 <> 'Headache')))  (cost=440 rows=3516) (actual time=0.018..1.7 rows=3814 loops=1)
    -> Table scan on PDR  (cost=440 rows=3516) (actual time=0.0167..1.4 rows=3814 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
1 row in set, 3 warnings (0.01 sec)
```

As evident from the screenshots, we added an index on the Keywords, FocusArea, and Question columns to try and improve the performance of this query. However, the cost of the query did not change. This is because the query uses a pattern like LIKE '%FocusArea%', which starts with a %. When this happens, MySQL cannot use the index efficiently because it has to check every row to see if the keyword appears anywhere in the text. This forces a full table scan. Because none of the schemes are improving our performance, we will be considering the default scheme.

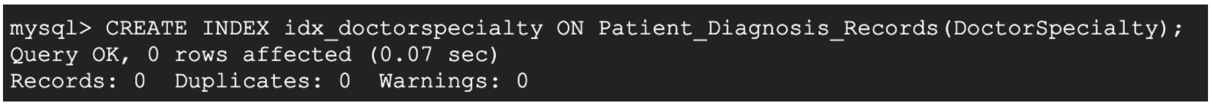
2) Indexing Query 2:

Performance without indexing:

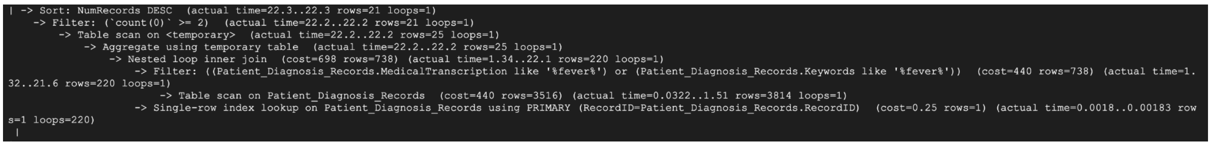


```
| -> Sort: NumRecords DESC  (actual time=23.2..23.2 rows=21 loops=1)
    -> Filter: (`count(0)` >= 2)  (actual time=23.2..23.2 rows=21 loops=1)
        -> Table scan on <temporary>  (actual time=23.2..23.2 rows=25 loops=1)
            -> Aggregate using temporary table  (actual time=23.2..23.2 rows=25 loops=1)
                -> Nested loop inner join  (cost=698 rows=738) (actual time=1.45..23 rows=220 loops=1)
                    -> Filter: ((Patient_Diagnosis_Records.MedicalTranscription like '%fever%') or (Patient_Diagnosis_Records.Keywords like '%fever%'))  (cost=440 rows=738) (actual time=1.
43..22.5 rows=220 loops=1)
                        -> Table scan on Patient_Diagnosis_Records  (cost=440 rows=3516) (actual time=0.0313..1.67 rows=3814 loops=1)
                    -> Single-row index lookup on Patient_Diagnosis_Records using PRIMARY (RecordID=Patient_Diagnosis_Records.RecordID)  (cost=0.25 rows=1) (actual time=0.00194..0.00197 ro
ws=1 loops=220)
 |
```

Performance after index on Patient_Diagnosis_Records(DoctorSpecialty)

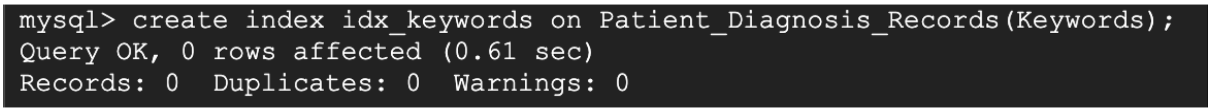


```
mysql> CREATE INDEX idx_doctorspecialty ON Patient_Diagnosis_Records(DoctorSpecialty);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

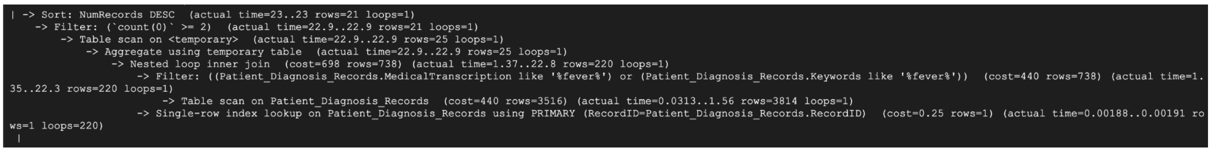


```
| -> Sort: NumRecords DESC  (actual time=22.3..22.3 rows=21 loops=1)
    -> Filter: (`count(0)` >= 2)  (actual time=22.2..22.2 rows=21 loops=1)
        -> Table scan on <temporary>  (actual time=22.2..22.2 rows=25 loops=1)
            -> Aggregate using temporary table  (actual time=22.2..22.2 rows=25 loops=1)
                -> Nested loop inner join  (cost=698 rows=738) (actual time=1.34..22.1 rows=220 loops=1)
                    -> Filter: ((Patient_Diagnosis_Records.MedicalTranscription like '%fever%') or (Patient_Diagnosis_Records.Keywords like '%fever%'))  (cost=440 rows=738) (actual time=1.
32..21.6 rows=220 loops=1)
                        -> Table scan on Patient_Diagnosis_Records  (cost=440 rows=3516) (actual time=0.0322..1.51 rows=3814 loops=1)
                    -> Single-row index lookup on Patient_Diagnosis_Records using PRIMARY (RecordID=Patient_Diagnosis_Records.RecordID)  (cost=0.25 rows=1) (actual time=0.0018..0.00183 row
s=1 loops=220)
 |
```

Performance after index on Patient_Diagnosis_Records(Keywords)

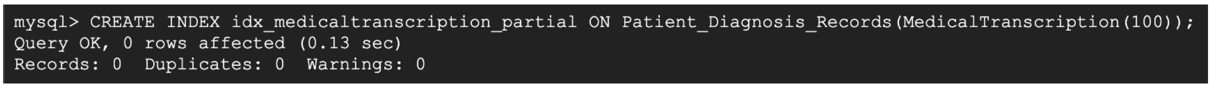


```
mysql> create index idx_keywords on Patient_Diagnosis_Records(Keywords);
Query OK, 0 rows affected (0.61 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

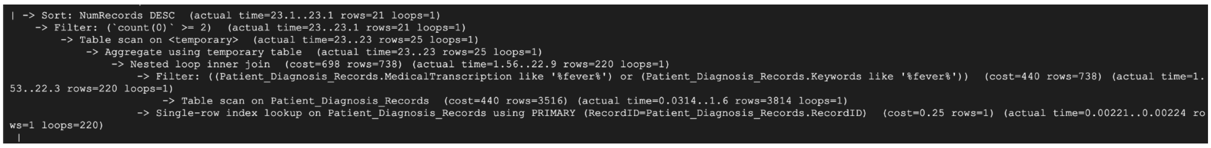


```
| -> Sort: NumRecords DESC  (actual time=23..23 rows=21 loops=1)
    -> Filter: (`count(0)` >= 2)  (actual time=22.9..22.9 rows=21 loops=1)
        -> Table scan on <temporary>  (actual time=22.9..22.9 rows=25 loops=1)
            -> Aggregate using temporary table  (actual time=22.9..22.9 rows=25 loops=1)
                -> Nested loop inner join  (cost=698 rows=738) (actual time=1.37..22.8 rows=220 loops=1)
                    -> Filter: ((Patient_Diagnosis_Records.MedicalTranscription like '%fever%') or (Patient_Diagnosis_Records.Keywords like '%fever%'))  (cost=440 rows=738) (actual time=1.
35..22.3 rows=220 loops=1)
                        -> Table scan on Patient_Diagnosis_Records  (cost=440 rows=3516) (actual time=0.0313..1.56 rows=3814 loops=1)
                    -> Single-row index lookup on Patient_Diagnosis_Records using PRIMARY (RecordID=Patient_Diagnosis_Records.RecordID)  (cost=0.25 rows=1) (actual time=0.00188..0.00191 ro
ws=1 loops=220)
 |
```

Performance after index on Patient_Diagnosis_Records(MedicalTranscription(100))



```
mysql> CREATE INDEX idx_medicaltranscription_partial ON Patient_Diagnosis_Records(MedicalTranscription(100));
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

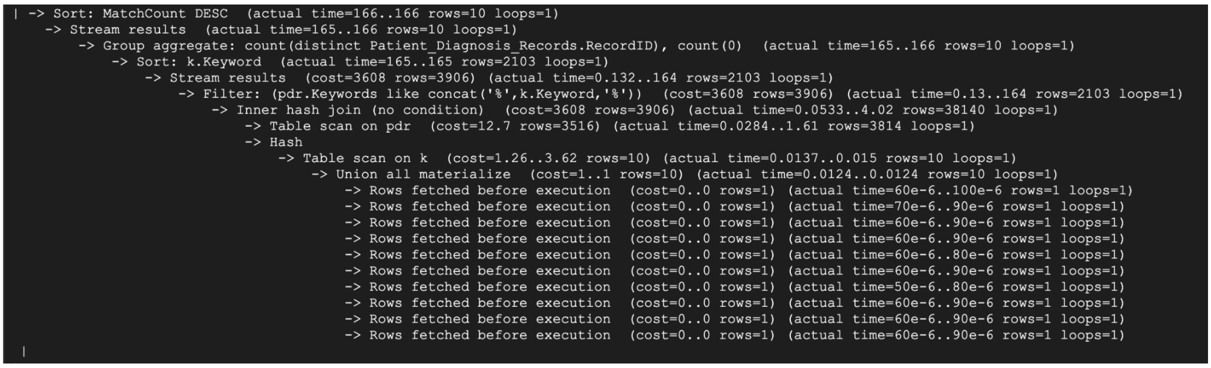


```
| -> Sort: NumRecords DESC  (actual time=23.1..23.1 rows=21 loops=1)
    -> Filter: (`count(0)` >= 2)  (actual time=23..23.1 rows=21 loops=1)
        -> Table scan on <temporary>  (actual time=23..23 rows=25 loops=1)
            -> Aggregate using temporary table  (actual time=23..23 rows=25 loops=1)
                -> Nested loop inner join  (cost=698 rows=738) (actual time=1.56..22.9 rows=220 loops=1)
                    -> Filter: ((Patient_Diagnosis_Records.MedicalTranscription like '%fever%') or (Patient_Diagnosis_Records.Keywords like '%fever%'))  (cost=440 rows=738) (actual time=1.
53..22.3 rows=220 loops=1)
                        -> Table scan on Patient_Diagnosis_Records  (cost=440 rows=3516) (actual time=0.0314..1.6 rows=3814 loops=1)
                    -> Single-row index lookup on Patient_Diagnosis_Records using PRIMARY (RecordID=Patient_Diagnosis_Records.RecordID)  (cost=0.25 rows=1) (actual time=0.00221..0.00224 ro
ws=1 loops=220)
 |
```

From the screenshots, you can see that we added indexing on DoctorSpecialty, Keywords, MedicalTranscription and see that the cost does not improve. This is because the query also uses a pattern like LIKE '%fever%', which starts with a %. MySQL cannot index efficiently because it has to check every row to see if the keyword appears anywhere in the text, essentially making the index redundant. Because none of the schemes are improving our performance, we will be considering the default scheme.

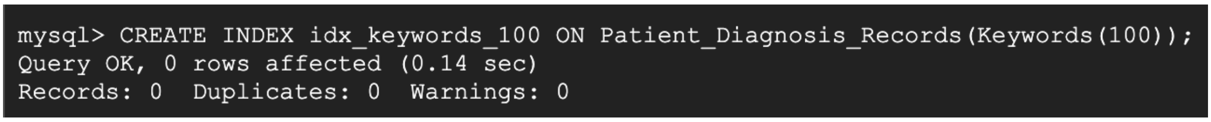
3) Indexing Query 3

Performance without indexing:

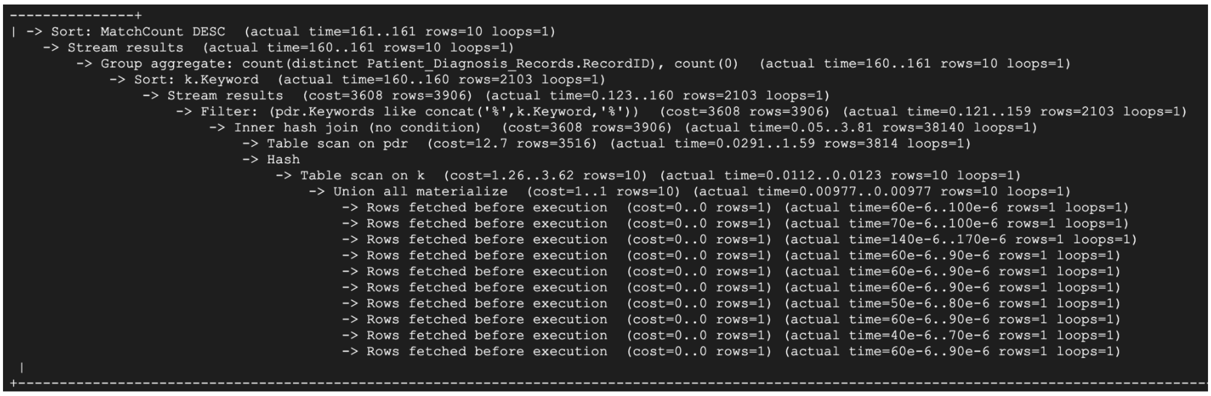


```
| -> Sort: MatchCount DESC  (actual time=166..166 rows=10 loops=1)
    -> Stream results  (actual time=165..166 rows=10 loops=1)
        -> Group aggregate: count(distinct Patient_Diagnosis_Records.RecordID), count(0)  (actual time=165..166 rows=10 loops=1)
            -> Sort: k.Keyword  (actual time=165..165 rows=2103 loops=1)
                -> Stream results  (cost=3608 rows=3906) (actual time=0.132..164 rows=2103 loops=1)
                    -> Filter: (pdr.Keywords like concat('%',k.Keyword,'%'))  (cost=3608 rows=3906) (actual time=0.13..164 rows=2103 loops=1)
                        -> Inner hash join (no condition)  (cost=3608 rows=3906) (actual time=0.0533..4.02 rows=38140 loops=1)
                            -> Table scan on pdr  (cost=12.7 rows=3516) (actual time=0.0284..1.61 rows=3814 loops=1)
                            -> Hash
                                -> Table scan on k  (cost=1.26..3.62 rows=10) (actual time=0.0137..0.015 rows=10 loops=1)
                                    -> Union all materialize  (cost=1..1 rows=10) (actual time=0.0124..0.0124 rows=10 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..100e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=70e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..80e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=50e-6..80e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
 |
```

Performance with index on Patient_Diagnosis_Records(Keywords(100)) - first 100 characters,

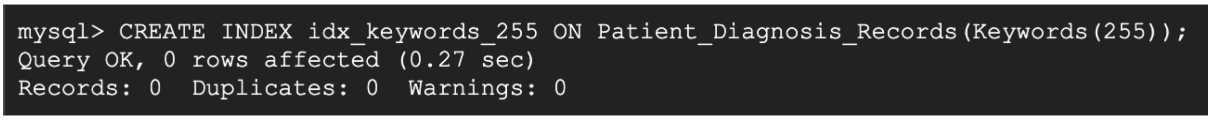


```
mysql> CREATE INDEX idx_keywords_100 ON Patient_Diagnosis_Records(Keywords(100));
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

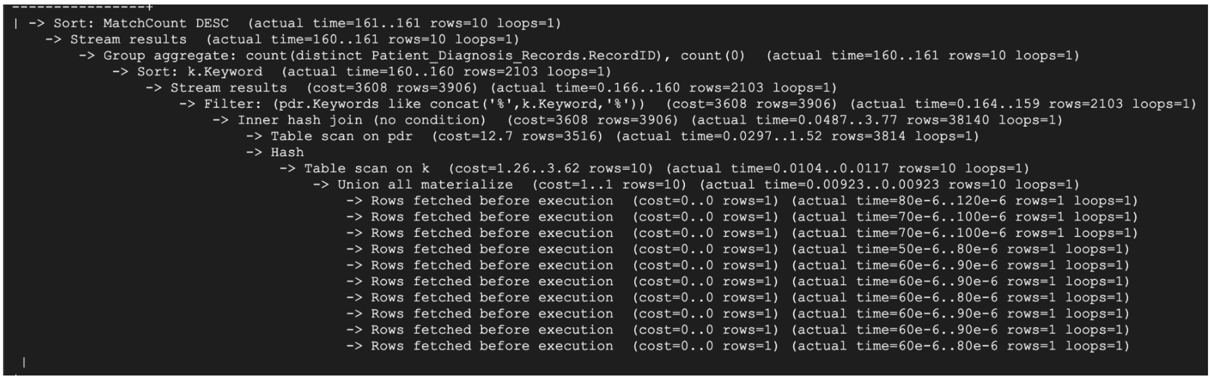


```
---------------+
| -> Sort: MatchCount DESC  (actual time=161..161 rows=10 loops=1)
    -> Stream results  (actual time=160..161 rows=10 loops=1)
        -> Group aggregate: count(distinct Patient_Diagnosis_Records.RecordID), count(0)  (actual time=160..161 rows=10 loops=1)
            -> Sort: k.Keyword  (actual time=160..160 rows=2103 loops=1)
                -> Stream results  (cost=3608 rows=3906) (actual time=0.123..160 rows=2103 loops=1)
                    -> Filter: (pdr.Keywords like concat('%',k.Keyword,'%'))  (cost=3608 rows=3906) (actual time=0.121..159 rows=2103 loops=1)
                        -> Inner hash join (no condition)  (cost=3608 rows=3906) (actual time=0.05..3.81 rows=38140 loops=1)
                            -> Table scan on pdr  (cost=12.7 rows=3516) (actual time=0.0291..1.59 rows=3814 loops=1)
                            -> Hash
                                -> Table scan on k  (cost=1.26..3.62 rows=10) (actual time=0.0112..0.0123 rows=10 loops=1)
                                    -> Union all materialize  (cost=1..1 rows=10) (actual time=0.00977..0.00977 rows=10 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..100e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=70e-6..100e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=140e-6..170e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=50e-6..80e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=40e-6..70e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Performance with index on Patient_Diagnosis_Records(Keywords(255)) - first 255 characters.

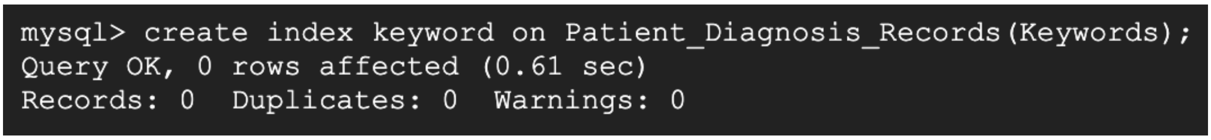


```
mysql> CREATE INDEX idx_keywords_255 ON Patient_Diagnosis_Records(Keywords(255));
Query OK, 0 rows affected (0.27 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

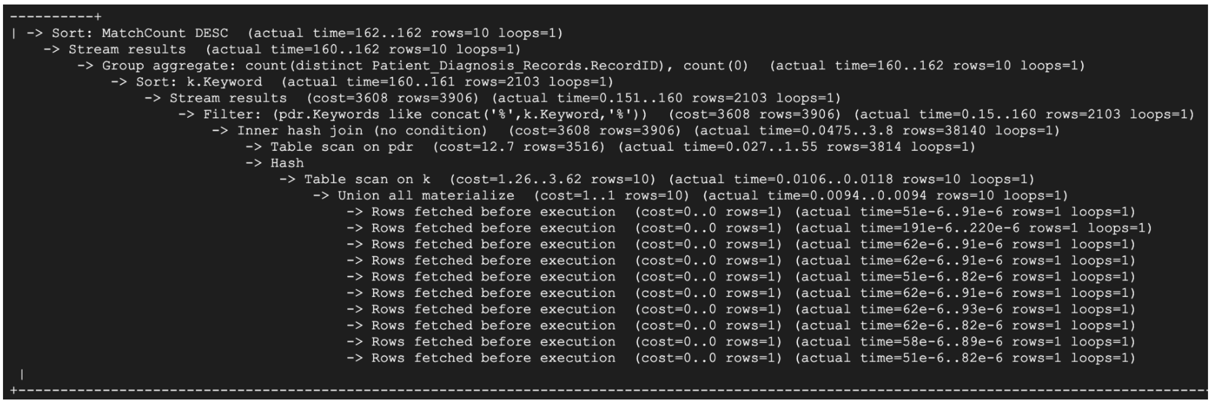


```
----------------+
| -> Sort: MatchCount DESC  (actual time=161..161 rows=10 loops=1)
    -> Stream results  (actual time=160..161 rows=10 loops=1)
        -> Group aggregate: count(distinct Patient_Diagnosis_Records.RecordID), count(0)  (actual time=160..161 rows=10 loops=1)
            -> Sort: k.Keyword  (actual time=160..160 rows=2103 loops=1)
                -> Stream results  (cost=3608 rows=3906) (actual time=0.166..160 rows=2103 loops=1)
                    -> Filter: (pdr.Keywords like concat('%',k.Keyword,'%'))  (cost=3608 rows=3906) (actual time=0.164..159 rows=2103 loops=1)
                        -> Inner hash join (no condition)  (cost=3608 rows=3906) (actual time=0.0487..3.77 rows=38140 loops=1)
                            -> Table scan on pdr  (cost=12.7 rows=3516) (actual time=0.0297..1.52 rows=3814 loops=1)
                            -> Hash
                                -> Table scan on k  (cost=1.26..3.62 rows=10) (actual time=0.0104..0.0117 rows=10 loops=1)
                                    -> Union all materialize  (cost=1..1 rows=10) (actual time=0.00923..0.00923 rows=10 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=80e-6..120e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=70e-6..100e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=70e-6..100e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=50e-6..80e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..80e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..90e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=60e-6..80e-6 rows=1 loops=1)
 |
```

Performance with index on Patient_Diagnosis_Records(Keywords) – all characters.



```
mysql> create index keyword on Patient_Diagnosis_Records(Keywords);
Query OK, 0 rows affected (0.61 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

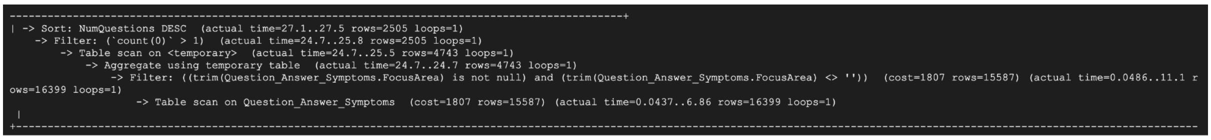


```
----------+
| -> Sort: MatchCount DESC  (actual time=162..162 rows=10 loops=1)
    -> Stream results  (actual time=160..162 rows=10 loops=1)
        -> Group aggregate: count(distinct Patient_Diagnosis_Records.RecordID), count(0)  (actual time=160..162 rows=10 loops=1)
            -> Sort: k.Keyword  (actual time=160..161 rows=2103 loops=1)
                -> Stream results  (cost=3608 rows=3906) (actual time=0.151..160 rows=2103 loops=1)
                    -> Filter: (pdr.Keywords like concat('%',k.Keyword,'%'))  (cost=3608 rows=3906) (actual time=0.15..160 rows=2103 loops=1)
                        -> Inner hash join (no condition)  (cost=3608 rows=3906) (actual time=0.0475..3.8 rows=38140 loops=1)
                            -> Table scan on pdr  (cost=12.7 rows=3516) (actual time=0.027..1.55 rows=3814 loops=1)
                            -> Hash
                                -> Table scan on k  (cost=1.26..3.62 rows=10) (actual time=0.0106..0.0118 rows=10 loops=1)
                                    -> Union all materialize  (cost=1..1 rows=10) (actual time=0.0094..0.0094 rows=10 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=51e-6..91e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=191e-6..220e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=62e-6..91e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=62e-6..91e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=51e-6..82e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=62e-6..91e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=62e-6..93e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=62e-6..82e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=58e-6..89e-6 rows=1 loops=1)
                                        -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=51e-6..82e-6 rows=1 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------
```

From the screenshots, you can see that we added indexing on Keywords for 3 different sizes and we see that the cost does not improve. This is because the query also uses a pattern like CONCAT LIKE, which starts with a %. MySQL cannot index efficiently because it has to check every row to see if the keyword appears anywhere in the text, essentially making the index redundant. Additionally, the format of the keywords data makes indexing difficult as there are not many shared values between records. As such even different sizes of Keywords were not able to reduce cost. Because none of the schemes are improving our performance, we will be considering the default scheme.

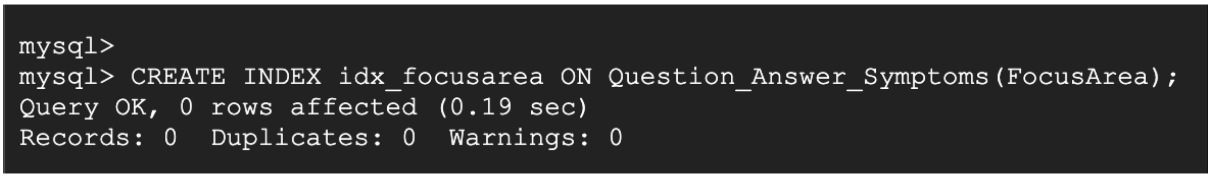
4) Indexing Query 4:

Performance without indexing:

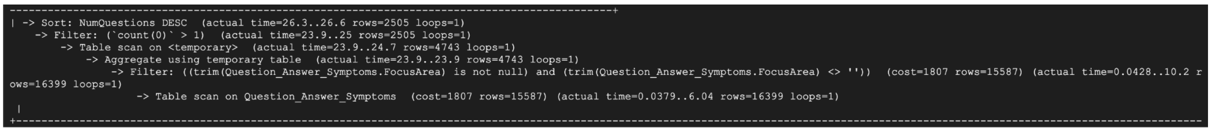


```
--------------------------------------------------------------------------------------------+
| -> Sort: NumQuestions DESC  (actual time=27.1..27.5 rows=2505 loops=1)
    -> Filter: (`count(0)` > 1)  (actual time=24.7..25.8 rows=2505 loops=1)
        -> Table scan on <temporary>  (actual time=24.7..25.5 rows=4743 loops=1)
            -> Aggregate using temporary table  (actual time=24.7..24.7 rows=4743 loops=1)
                -> Filter: ((trim(Question_Answer_Symptoms.FocusArea) is not null) and (trim(Question_Answer_Symptoms.FocusArea) <> ''))  (cost=1807 rows=15587) (actual time=0.0486..11.1 r
ows=16399 loops=1)
                    -> Table scan on Question_Answer_Symptoms  (cost=1807 rows=15587) (actual time=0.0437..6.86 rows=16399 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Performance with index on Question_Answer_Symptoms(FocusArea)

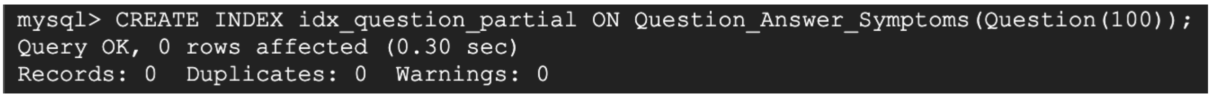


```
mysql>
mysql> CREATE INDEX idx_focusarea ON Question_Answer_Symptoms(FocusArea);
Query OK, 0 rows affected (0.19 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

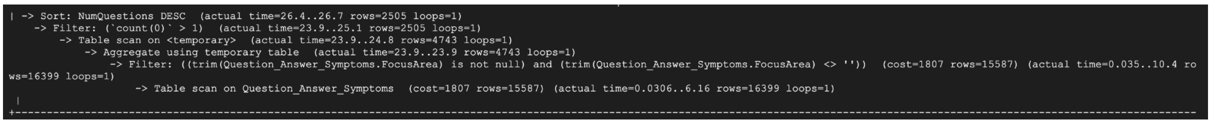


```
------------------------------------------------------------------------------------------+
| -> Sort: NumQuestions DESC  (actual time=26.3..26.6 rows=2505 loops=1)
    -> Filter: (`count(0)` > 1)  (actual time=23.9..25 rows=2505 loops=1)
        -> Table scan on <temporary>  (actual time=23.9..24.7 rows=4743 loops=1)
            -> Aggregate using temporary table  (actual time=23.9..23.9 rows=4743 loops=1)
                -> Filter: ((trim(Question_Answer_Symptoms.FocusArea) is not null) and (trim(Question_Answer_Symptoms.FocusArea) <> ''))  (cost=1807 rows=15587) (actual time=0.0428..10.2 r
ows=16399 loops=1)
                    -> Table scan on Question_Answer_Symptoms  (cost=1807 rows=15587) (actual time=0.0379..6.04 rows=16399 loops=1)
 |
+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Performance with index on Question_Answer_Symptoms(Question(100))

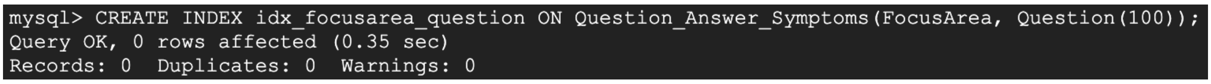


```
mysql> CREATE INDEX idx_question_partial ON Question_Answer_Symptoms(Question(100));
Query OK, 0 rows affected (0.30 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

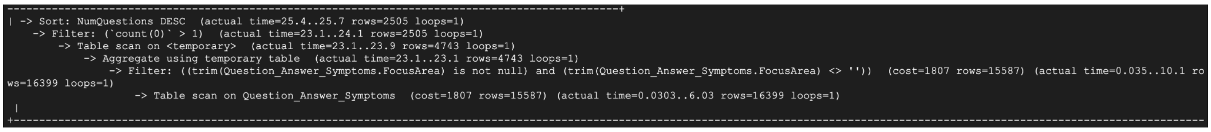


```
| -> Sort: NumQuestions DESC  (actual time=26.4..26.7 rows=2505 loops=1)
    -> Filter: (`count(0)` > 1)  (actual time=23.9..25.1 rows=2505 loops=1)
        -> Table scan on <temporary>  (actual time=23.9..24.8 rows=4743 loops=1)
            -> Aggregate using temporary table  (actual time=23.9..23.9 rows=4743 loops=1)
                -> Filter: ((trim(Question_Answer_Symptoms.FocusArea) is not null) and (trim(Question_Answer_Symptoms.FocusArea) <> ''))  (cost=1807 rows=15587) (actual time=0.035..10.4 ro
ws=16399 loops=1)
                    -> Table scan on Question_Answer_Symptoms  (cost=1807 rows=15587) (actual time=0.0306..6.16 rows=16399 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Performance with composite index on Question_Answer_Symptoms(FocusArea, Question(100))



```
mysql> CREATE INDEX idx_focusarea_question ON Question_Answer_Symptoms(FocusArea, Question(100));
Query OK, 0 rows affected (0.35 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
-------------------------------------------------------------------------------------------------+
| -> Sort: NumQuestions DESC  (actual time=25.4..25.7 rows=2505 loops=1)
    -> Filter: (`count(0)` > 1)  (actual time=23.1..24.1 rows=2505 loops=1)
        -> Table scan on <temporary>  (actual time=23.1..23.9 rows=4743 loops=1)
            -> Aggregate using temporary table  (actual time=23.1..23.1 rows=4743 loops=1)
                -> Filter: ((trim(Question_Answer_Symptoms.FocusArea) is not null) and (trim(Question_Answer_Symptoms.FocusArea) <> ''))  (cost=1807 rows=15587) (actual time=0.035..10.1 ro
ws=16399 loops=1)
                    -> Table scan on Question_Answer_Symptoms  (cost=1807 rows=15587) (actual time=0.0303..6.03 rows=16399 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

In the screenshots, we have indexed our query on FocusArea, Question, and the pair of FocusArea and Question and see that the performance of the query does not improve. This is because the TRIM function is used across different places in the query. This means that even if we index FocusArea, the TRIM usage in many parts of the query disables its effect. However, we could optimize this a little if we are able to trim the focus area in some data preprocessing, which we will do in the next stage while building the application. Because none of the schemes are improving our performance, we will be considering the default scheme for now.