DATABASE DESIGN

(PT1: STAGE 3)

PART 1:

1. The five tables created are Users, Diseases, Symptoms, Diagnosis and Medicines (with another table possible_diseases which is the output of the advanced query 1)

2. CREATE TABLE Users (UserID int NOT NULL PRIMARY KEY, FirstName Varchar(100) NOT NULL, LastName Varchar(100) NOT NULL, Height float, Weight float, Sex Varchar(1), Age int, Race Varchar(100), Email Varchar(100));

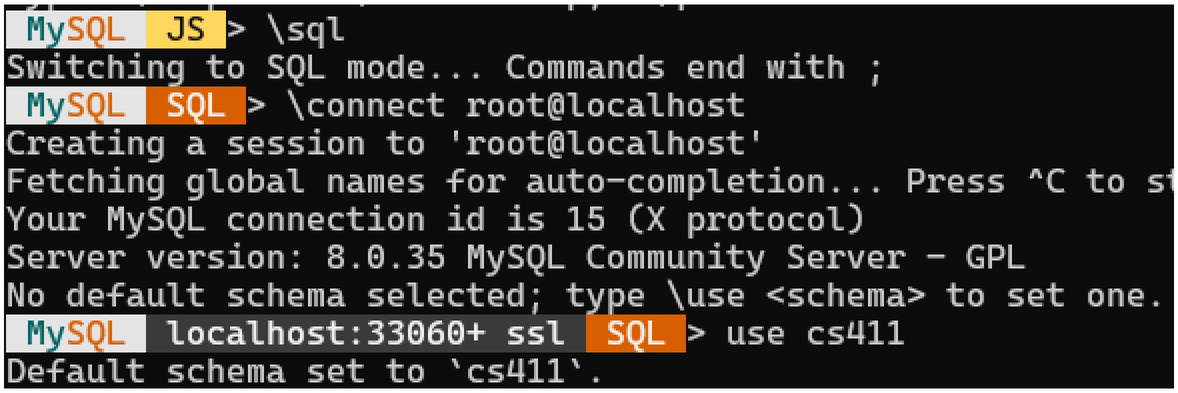
   CREATE TABLE Diseases (DiseaseID int NOT NULL, DiseaseName Varchar(100) NOT NULL PRIMARY KEY, Rarity int);

   CREATE TABLE Symptoms (MatchID int NOT NULL PRIMARY KEY, SymptomName Varchar(100) NOT NULL, DiseaseName Varchar(100), Severity int, FOREIGN KEY(DiseaseName) REFERENCES diseases(DiseaseName));

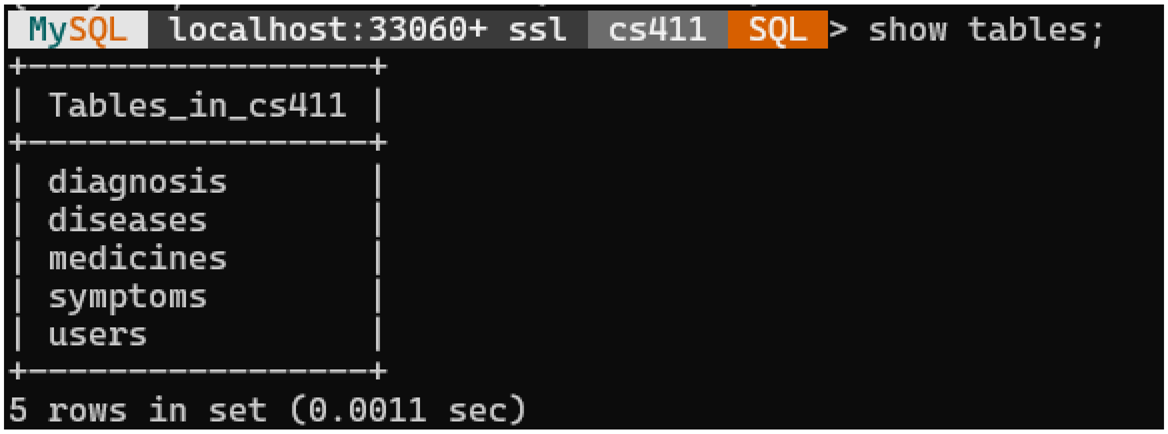
   CREATE TABLE Diagnosis (DiagnosisID int AUTO_INCREMENT NOT NULL PRIMARY KEY, UserID int NOT NULL, s1 Varchar(100) NOT NULL, s2 Varchar(100), s3 Varchar(100), DiseaseName Varchar(100), FOREIGN KEY (UserID) REFERENCES users(UserID), FOREIGN KEY (DiseaseName) REFERENCES diseases(DiseaseName));

   CREATE TABLE Medicines (MedicineID int NOT NULL PRIMARY KEY, MedicineName Varchar(100) NOT NULL, Composition Varchar(500) NOT NULL, Uses Varchar(500) NOT NULL, SideEffects Varchar(1000) NOT NULL, ImageURL Varchar(500) NOT NULL, Manufacturer Varchar(300) NOT NULL, GoodReviewsPercent int NOT NULL, AvgReviewsPercent int NOT NULL, BadReviewsPercent int NOT NULL);

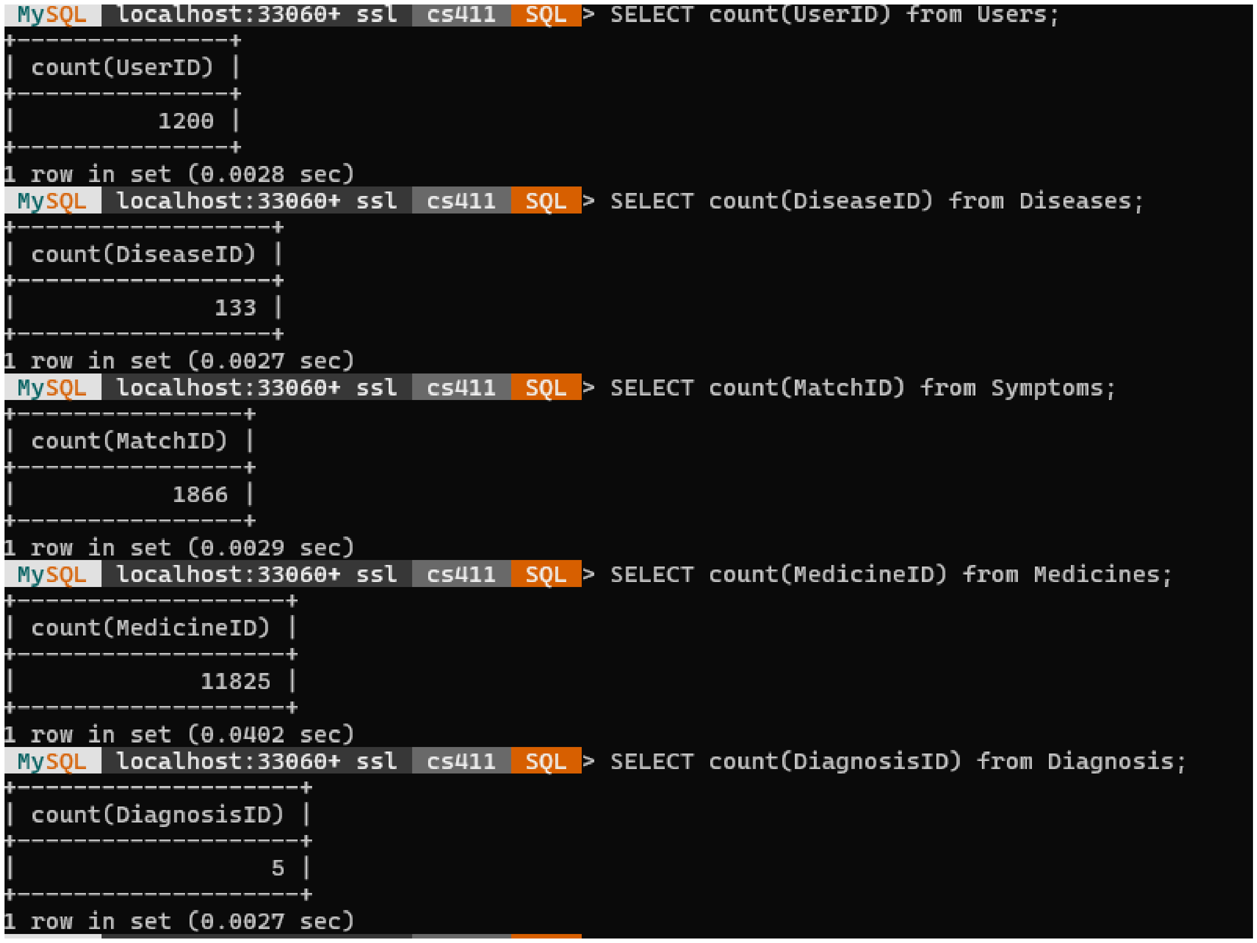
   Screenshots of local connection:



```
 MySQL  JS > \sql
Switching to SQL mode... Commands end with ;
 MySQL  SQL > \connect root@localhost
Creating a session to 'root@localhost'
Fetching global names for auto-completion... Press ^C to st
Your MySQL connection id is 15 (X protocol)
Server version: 8.0.35 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
 MySQL  localhost:33060+ ssl  SQL > use cs411
Default schema set to `cs411`.
```



```
 MySQL  localhost:33060+ ssl  cs411  SQL > show tables;
+-----------------+
| Tables_in_cs411 |
+-----------------+
| diagnosis       |
| diseases        |
| medicines       |
| symptoms        |
| users           |
+-----------------+
5 rows in set (0.0011 sec)
```



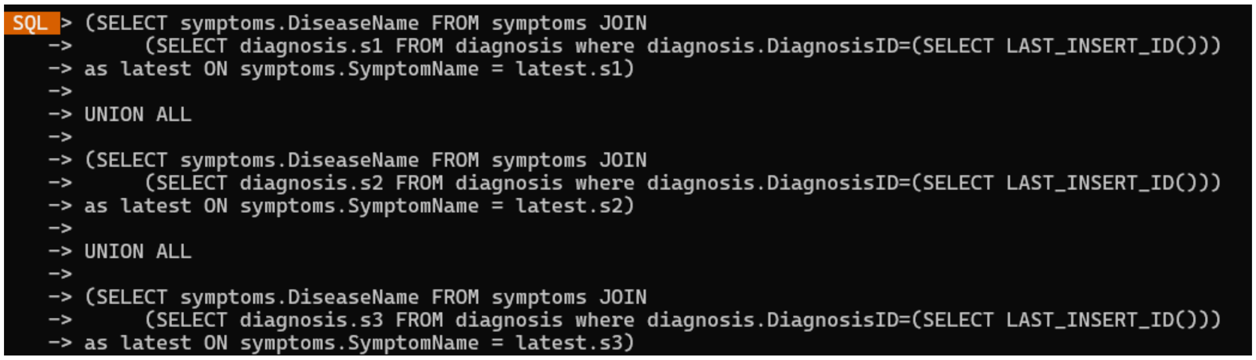
```
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT count(UserID) from Users;
+---------------+
| count(UserID) |
+---------------+
|          1200 |
+---------------+
1 row in set (0.0028 sec)
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT count(DiseaseID) from Diseases;
+------------------+
| count(DiseaseID) |
+------------------+
|              133 |
+------------------+
1 row in set (0.0027 sec)
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT count(MatchID) from Symptoms;
+----------------+
| count(MatchID) |
+----------------+
|           1866 |
+----------------+
1 row in set (0.0029 sec)
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT count(MedicineID) from Medicines;
+-------------------+
| count(MedicineID) |
+-------------------+
|             11825 |
+-------------------+
1 row in set (0.0402 sec)
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT count(DiagnosisID) from Diagnosis;
+--------------------+
| count(DiagnosisID) |
+--------------------+
|                  5 |
+--------------------+
1 row in set (0.0027 sec)
```

3. To fill the tables, we used CSV files for Users, Diseases, Symptoms and Medicines tables (which have been added to the github repo) and hardcoded the Diagnosis table (since it would contain the userID and symptoms given to us when a user wants to find out their potential disease). The tables Users, Diseases and Symptoms have more than 100 rows each. The data from Diseases and Symptoms were made using various real datasets from Kaggle while Users were auto-generated. The code for inserting the data from the CSV files was:

   LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/***THE CSV FILE***' INTO TABLE ***NAME OF THE TABLE*** FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;

   To fill the Diagnosis table, we used a random UserID and upto 3 symptoms (but at least 1)

4. Query 1:

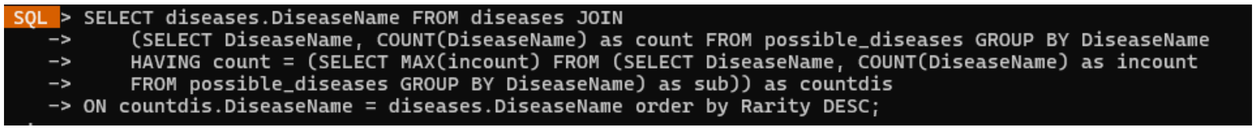


```
SQL > (SELECT symptoms.DiseaseName FROM symptoms JOIN
   ->      (SELECT diagnosis.s1 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
   -> as latest ON symptoms.SymptomName = latest.s1)
   ->
   -> UNION ALL
   ->
   -> (SELECT symptoms.DiseaseName FROM symptoms JOIN
   ->      (SELECT diagnosis.s2 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
   -> as latest ON symptoms.SymptomName = latest.s2)
   ->
   -> UNION ALL
   ->
   -> (SELECT symptoms.DiseaseName FROM symptoms JOIN
   ->      (SELECT diagnosis.s3 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
   -> as latest ON symptoms.SymptomName = latest.s3)
```

This query takes upto 3 symptoms from the most recent entry in the Diagnosis table and gives a combined list of all possible diseases that each symptom may suggest. We save this query in a seperate table ‘possible_diseases’ using:

CREATE TABLE possible_diseases AS (***Query 1***)

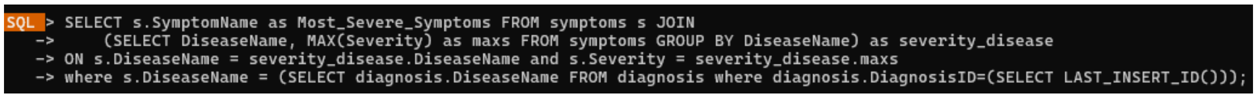
Query 2:



```
SQL > SELECT diseases.DiseaseName FROM diseases JOIN
   ->     (SELECT DiseaseName, COUNT(DiseaseName) as count FROM possible_diseases GROUP BY DiseaseName
   ->     HAVING count = (SELECT MAX(incount) FROM (SELECT DiseaseName, COUNT(DiseaseName) as incount
   ->     FROM possible_diseases GROUP BY DiseaseName) as sub)) as countdis
   -> ON countdis.DiseaseName = diseases.DiseaseName order by Rarity DESC;
```

This query gives us the most likely diseases from all possible diseases (by seeing which disease is repeated the most) and since they are ordered by descending rarity, the predicted disease that is entered in the Diagnosis table is the first one.Then we update our diagnosis table with the predicted disease:
UPDATE Diagnosis SET DiseaseName = ***Query 2 LIMIT 1*** WHERE DiagnosisID = (SELECT LAST_INSERT_ID());

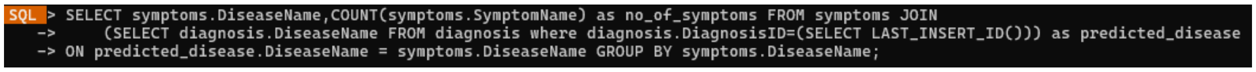
Query 3:



```
SQL > SELECT s.SymptomName as Most_Severe_Symptoms FROM symptoms s JOIN
   ->     (SELECT DiseaseName, MAX(Severity) as maxs FROM symptoms GROUP BY DiseaseName) as severity_disease
   -> ON s.DiseaseName = severity_disease.DiseaseName and s.Severity = severity_disease.maxs
   -> where s.DiseaseName = (SELECT diagnosis.DiseaseName FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()));
```

This query can tell the user what are the most severe symptoms associated with the predicted disease (to warn them).

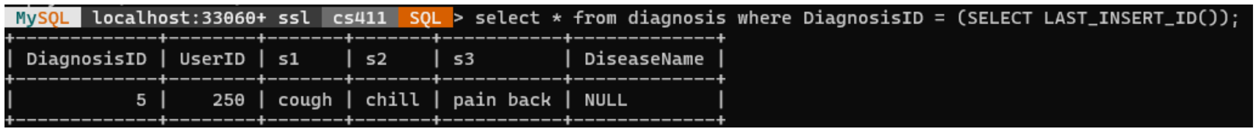
Query 4:



```
SQL > SELECT symptoms.DiseaseName,COUNT(symptoms.SymptomName) as no_of_symptoms FROM symptoms JOIN
   ->     (SELECT diagnosis.DiseaseName FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID())) as predicted_disease
   -> ON predicted_disease.DiseaseName = symptoms.DiseaseName GROUP BY symptoms.DiseaseName;
```

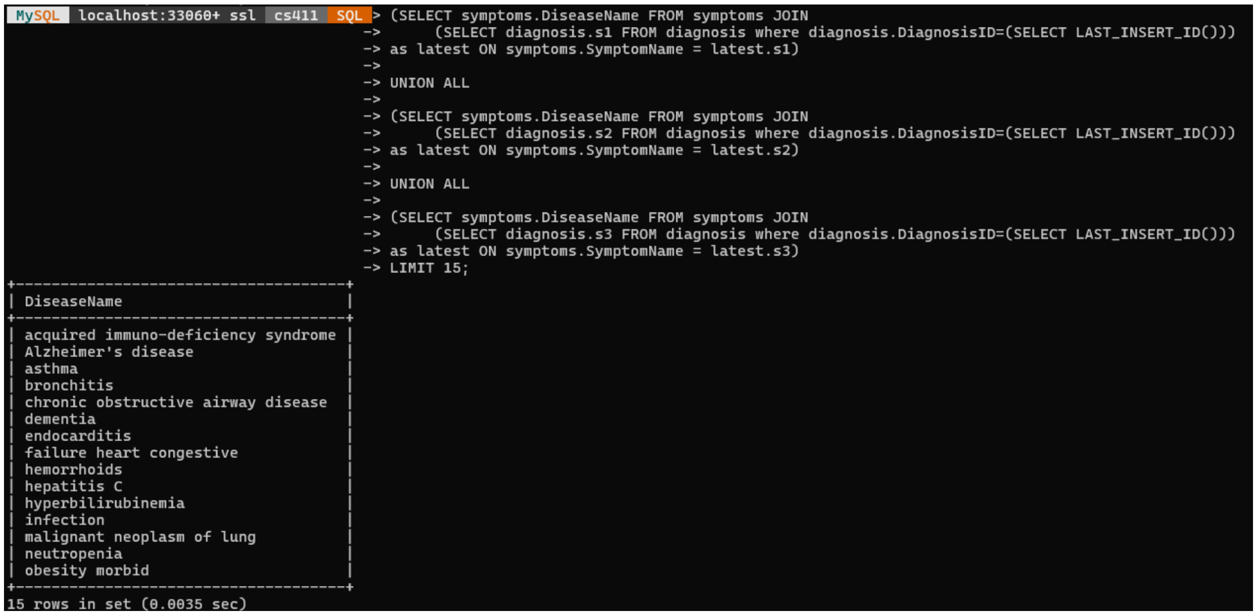
This query can tell the user how many more symptoms their predicted disease has.

5. Here, first the user enters a UserID and symptoms (i.e. most recent entry of the Diagnosis table):



```
 MySQL  localhost:33060+ ssl  cs411  SQL > select * from diagnosis where DiagnosisID = (SELECT LAST_INSERT_ID());
+-------------+--------+-------+-------+-----------+-------------+
| DiagnosisID | UserID | s1    | s2    | s3        | DiseaseName |
+-------------+--------+-------+-------+-----------+-------------+
|           5 |    250 | cough | chill | pain back | NULL        |
+-------------+--------+-------+-------+-----------+-------------+
```

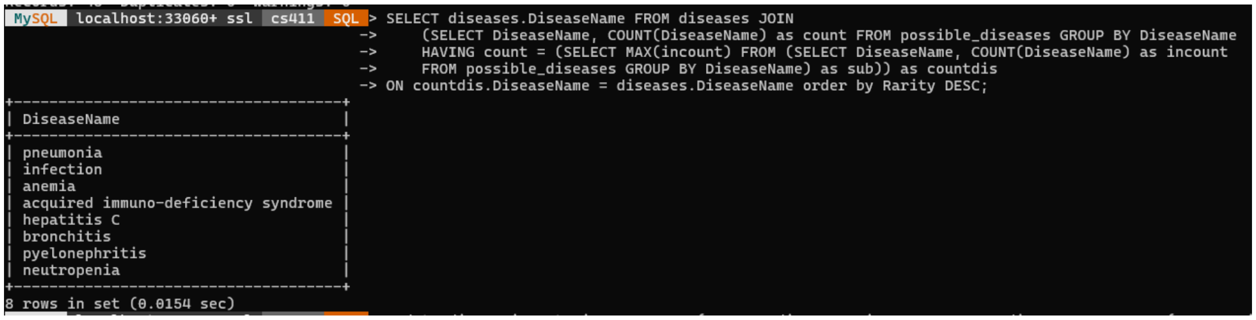
Output of Query 1 (with LIMIT 15):



```
 MySQL  localhost:33060+ ssl  cs411  SQL > (SELECT symptoms.DiseaseName FROM symptoms JOIN
                                        ->      (SELECT diagnosis.s1 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
                                        -> as latest ON symptoms.SymptomName = latest.s1)
                                        ->
                                        -> UNION ALL
                                        ->
                                        -> (SELECT symptoms.DiseaseName FROM symptoms JOIN
                                        ->      (SELECT diagnosis.s2 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
                                        -> as latest ON symptoms.SymptomName = latest.s2)
                                        ->
                                        -> UNION ALL
                                        ->
                                        -> (SELECT symptoms.DiseaseName FROM symptoms JOIN
                                        ->      (SELECT diagnosis.s3 FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()))
                                        -> as latest ON symptoms.SymptomName = latest.s3)
                                        -> LIMIT 15;
+--------------------------------------+
| DiseaseName                          |
+--------------------------------------+
| acquired immuno-deficiency syndrome |
| Alzheimer's disease                  |
| asthma                               |
| bronchitis                           |
| chronic obstructive airway disease   |
| dementia                             |
| endocarditis                         |
| failure heart congestive             |
| hemorrhoids                          |
| hepatitis C                          |
| hyperbilirubinemia                   |
| infection                            |
| malignant neoplasm of lung           |
| neutropenia                          |
| obesity morbid                       |
+--------------------------------------+
15 rows in set (0.0035 sec)
```

Like explained above, Query 1 is saved in possible_diseases

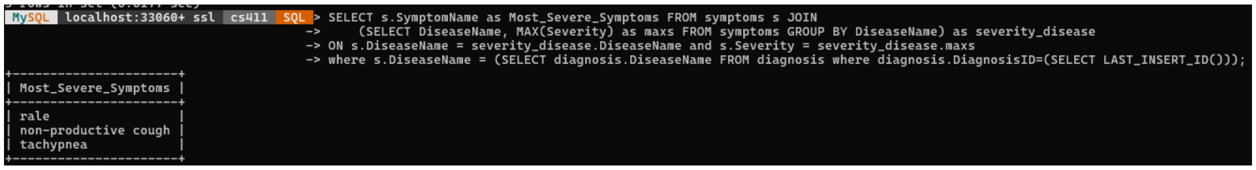
Output of Query 2 (Output is less than 15):



```
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT diseases.DiseaseName FROM diseases JOIN
                                       ->     (SELECT DiseaseName, COUNT(DiseaseName) as count FROM possible_diseases GROUP BY DiseaseName
                                       ->     HAVING count = (SELECT MAX(incount) FROM (SELECT DiseaseName, COUNT(DiseaseName) as incount
                                       ->     FROM possible_diseases GROUP BY DiseaseName) as sub)) as countdis
                                       -> ON countdis.DiseaseName = diseases.DiseaseName order by Rarity DESC;
+--------------------------------------+
| DiseaseName                          |
+--------------------------------------+
| pneumonia                            |
| infection                            |
| anemia                               |
| acquired immuno-deficiency syndrome  |
| hepatitis C                          |
| bronchitis                           |
| pyelonephritis                       |
| neutropenia                          |
+--------------------------------------+
8 rows in set (0.0154 sec)
```

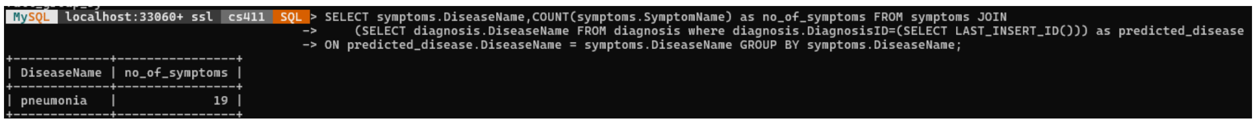
Predicted disease is the most common Disease of this list which is the first one (as they are in descending order of rarity), so Diagnosis table is updated.

Output of Query 3 (Output is less than 15):



```
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT s.SymptomName as Most_Severe_Symptoms FROM symptoms s JOIN
                                       ->     (SELECT DiseaseName, MAX(Severity) as maxs FROM symptoms GROUP BY DiseaseName) as severity_disease
                                       -> ON s.DiseaseName = severity_disease.DiseaseName and s.Severity = severity_disease.maxs
                                       -> where s.DiseaseName = (SELECT diagnosis.DiseaseName FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID()));
+----------------------+
| Most_Severe_Symptoms |
+----------------------+
| rale                 |
| non-productive cough |
| tachypnea            |
+----------------------+
```

Output of Query 4 (Output is always one row):

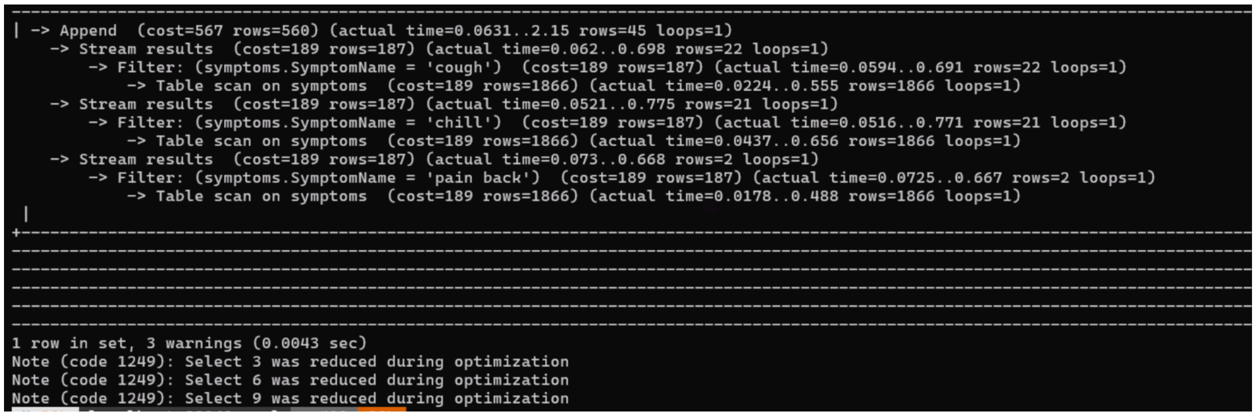


```
MySQL  localhost:33060+ ssl  cs411  SQL > SELECT symptoms.DiseaseName,COUNT(symptoms.SymptomName) as no_of_symptoms FROM symptoms JOIN
                                       ->     (SELECT diagnosis.DiseaseName FROM diagnosis where diagnosis.DiagnosisID=(SELECT LAST_INSERT_ID())) as predicted_disease
                                       -> ON predicted_disease.DiseaseName = symptoms.DiseaseName GROUP BY symptoms.DiseaseName;
+-------------+----------------+
| DiseaseName | no_of_symptoms |
+-------------+----------------+
| pneumonia   |             19 |
+-------------+----------------+
```

PART 2:

- Query 1 :

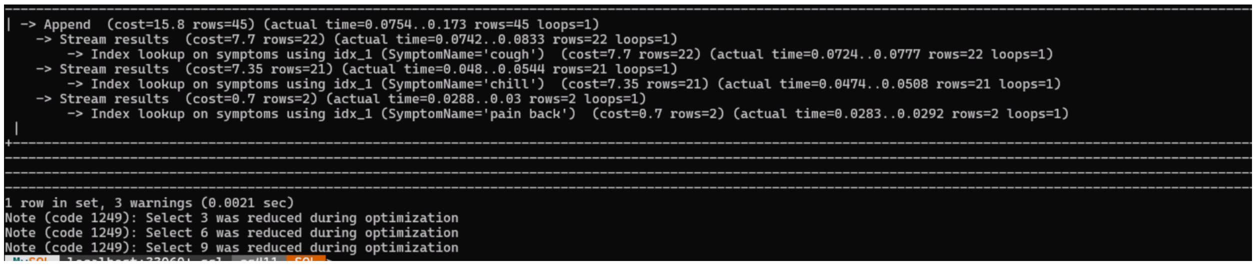
Default indexing -



```
| -> Append  (cost=567 rows=560) (actual time=0.0631..2.15 rows=45 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.062..0.698 rows=22 loops=1)
        -> Filter: (symptoms.SymptomName = 'cough')  (cost=189 rows=187) (actual time=0.0594..0.691 rows=22 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0224..0.555 rows=1866 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.0521..0.775 rows=21 loops=1)
        -> Filter: (symptoms.SymptomName = 'chill')  (cost=189 rows=187) (actual time=0.0516..0.771 rows=21 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0437..0.656 rows=1866 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.073..0.668 rows=2 loops=1)
        -> Filter: (symptoms.SymptomName = 'pain back')  (cost=189 rows=187) (actual time=0.0725..0.667 rows=2 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0178..0.488 rows=1866 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------

1 row in set, 3 warnings (0.0043 sec)
Note (code 1249): Select 3 was reduced during optimization
Note (code 1249): Select 6 was reduced during optimization
Note (code 1249): Select 9 was reduced during optimization
```

Indexing Symptom(SymptomName) -

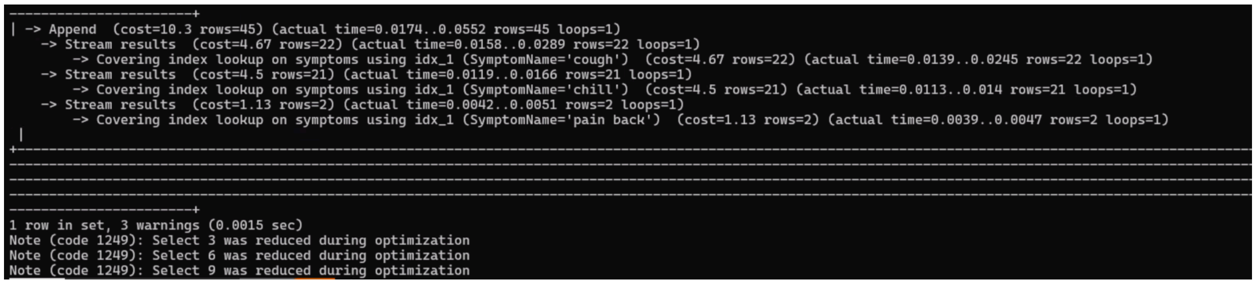


```
| -> Append  (cost=15.8 rows=45) (actual time=0.0754..0.173 rows=45 loops=1)
    -> Stream results  (cost=7.7 rows=22) (actual time=0.0742..0.0833 rows=22 loops=1)
        -> Index lookup on symptoms using idx_1 (SymptomName='cough')  (cost=7.7 rows=22) (actual time=0.0724..0.0777 rows=22 loops=1)
    -> Stream results  (cost=7.35 rows=21) (actual time=0.048..0.0544 rows=21 loops=1)
        -> Index lookup on symptoms using idx_1 (SymptomName='chill')  (cost=7.35 rows=21) (actual time=0.0474..0.0508 rows=21 loops=1)
    -> Stream results  (cost=0.7 rows=2) (actual time=0.0288..0.03 rows=2 loops=1)
        -> Index lookup on symptoms using idx_1 (SymptomName='pain back')  (cost=0.7 rows=2) (actual time=0.0283..0.0292 rows=2 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------

1 row in set, 3 warnings (0.0021 sec)
Note (code 1249): Select 3 was reduced during optimization
Note (code 1249): Select 6 was reduced during optimization
Note (code 1249): Select 9 was reduced during optimization
```

This gives us a performance gain, changing the total cost from 567 to 15.8 since the cost of finding each of the three symptoms in the join condition with the Symptom table is drastically reduced due to this indexing.

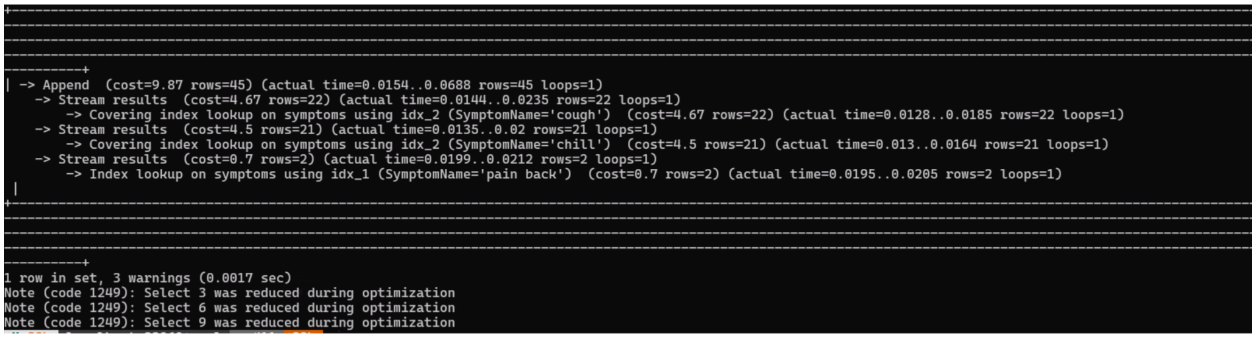
Indexing Symptom(SymptomName, DiseaseName) output-



```
-----------------------+
| -> Append  (cost=10.3 rows=45) (actual time=0.0174..0.0552 rows=45 loops=1)
    -> Stream results  (cost=4.67 rows=22) (actual time=0.0158..0.0289 rows=22 loops=1)
        -> Covering index lookup on symptoms using idx_1 (SymptomName='cough')  (cost=4.67 rows=22) (actual time=0.0139..0.0245 rows=22 loops=1)
    -> Stream results  (cost=4.5 rows=21) (actual time=0.0119..0.0166 rows=21 loops=1)
        -> Covering index lookup on symptoms using idx_1 (SymptomName='chill')  (cost=4.5 rows=21) (actual time=0.0113..0.014 rows=21 loops=1)
    -> Stream results  (cost=1.13 rows=2) (actual time=0.0042..0.0051 rows=2 loops=1)
        -> Covering index lookup on symptoms using idx_1 (SymptomName='pain back')  (cost=1.13 rows=2) (actual time=0.0039..0.0047 rows=2 loops=1)
 |
+--------------------------------------------------------------------------------------------------------------------------------
-----------------------+
1 row in set, 3 warnings (0.0015 sec)
Note (code 1249): Select 3 was reduced during optimization
Note (code 1249): Select 6 was reduced during optimization
Note (code 1249): Select 9 was reduced during optimization
```

This too gives us a performance gain, changing the total cost from 567 to 10.3 since the cost of listing all the diseases linked to the 3 symptoms and unioning them is further reduced due to this indexing.

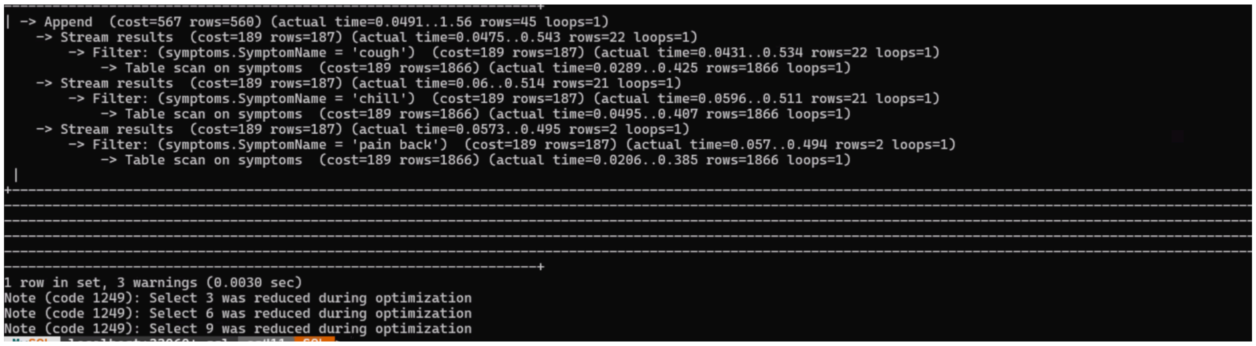
Indexing Symptom(symptomName) and indexing on Symptom(SymptomName, DiseaseName) -



```
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
----------+
| -> Append  (cost=9.87 rows=45) (actual time=0.0154..0.0688 rows=45 loops=1)
    -> Stream results  (cost=4.67 rows=22) (actual time=0.0144..0.0235 rows=22 loops=1)
        -> Covering index lookup on symptoms using idx_2 (SymptomName='cough')  (cost=4.67 rows=22) (actual time=0.0128..0.0185 rows=22 loops=1)
    -> Stream results  (cost=4.5 rows=21) (actual time=0.0135..0.02 rows=21 loops=1)
        -> Covering index lookup on symptoms using idx_2 (SymptomName='chill')  (cost=4.5 rows=21) (actual time=0.013..0.0164 rows=21 loops=1)
    -> Stream results  (cost=0.7 rows=2) (actual time=0.0199..0.0212 rows=2 loops=1)
        -> Index lookup on symptoms using idx_1 (SymptomName='pain back')  (cost=0.7 rows=2) (actual time=0.0195..0.0205 rows=2 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
----------+
1 row in set, 3 warnings (0.0017 sec)
Note (code 1249): Select 3 was reduced during optimization
Note (code 1249): Select 6 was reduced during optimization
Note (code 1249): Select 9 was reduced during optimization
```

This gives us the most performance gain due to a combination of the 2 factors above.

Indexing Diagnosis(s1, s2, s3) -



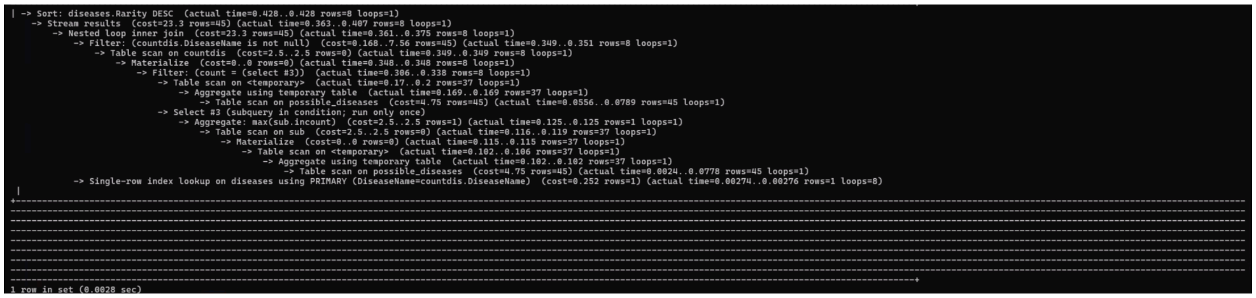
```
------------------------------------------------------------------+
| -> Append  (cost=567 rows=560) (actual time=0.0491..1.56 rows=45 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.0475..0.543 rows=22 loops=1)
        -> Filter: (symptoms.SymptomName = 'cough')  (cost=189 rows=187) (actual time=0.0431..0.534 rows=22 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0289..0.425 rows=1866 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.06..0.514 rows=21 loops=1)
        -> Filter: (symptoms.SymptomName = 'chill')  (cost=189 rows=187) (actual time=0.0596..0.511 rows=21 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0495..0.407 rows=1866 loops=1)
    -> Stream results  (cost=189 rows=187) (actual time=0.0573..0.495 rows=2 loops=1)
        -> Filter: (symptoms.SymptomName = 'pain back')  (cost=189 rows=187) (actual time=0.057..0.494 rows=2 loops=1)
            -> Table scan on symptoms  (cost=189 rows=1866) (actual time=0.0206..0.385 rows=1866 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------+
1 row in set, 3 warnings (0.0030 sec)
Note (code 1249): Select 3 was reduced during optimization
Note (code 1249): Select 6 was reduced during optimization
Note (code 1249): Select 9 was reduced during optimization
```

This does not cause any change in performance since we only access the most recent entry from the diagnosis table so since there is only one record available anyway, indexing it does not affect performance.

Final Index Design: The final index design we selected was Indexing Symptom(symptomName) and indexing on Symptom(SymptomName, DiseaseName) as cost of finding each of the three symptoms in the join condition is most reduced for this design.

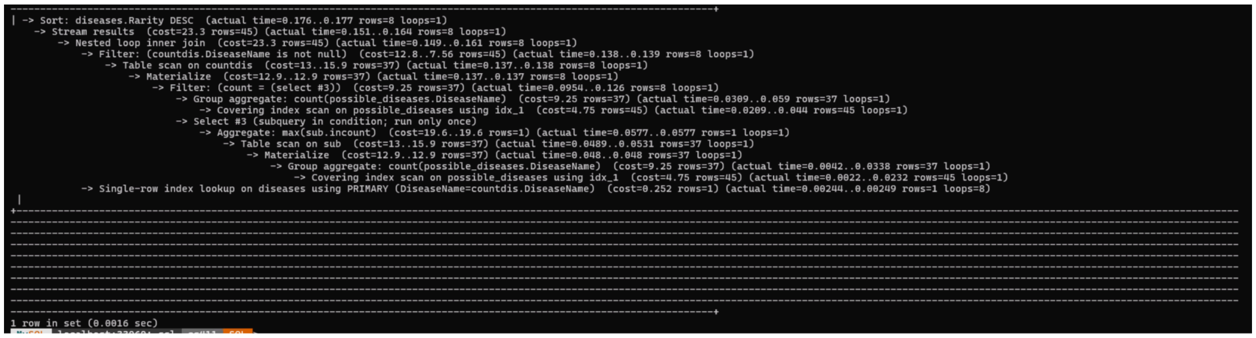
- Query 2 :

Default indexing -



```
| -> Sort: diseases.Rarity DESC  (actual time=0.428..0.428 rows=8 loops=1)
    -> Stream results  (cost=23.3 rows=45) (actual time=0.363..0.407 rows=8 loops=1)
        -> Nested loop inner join  (cost=23.3 rows=45) (actual time=0.361..0.375 rows=8 loops=1)
            -> Filter: (countdis.DiseaseName is not null)  (cost=0.168..7.56 rows=45) (actual time=0.349..0.351 rows=8 loops=1)
                -> Table scan on countdis  (cost=2.5..2.5 rows=0) (actual time=0.349..0.349 rows=8 loops=1)
                    -> Materialize  (cost=0..0 rows=0) (actual time=0.348..0.348 rows=8 loops=1)
                        -> Filter: (count = (select #3))  (actual time=0.306..0.338 rows=8 loops=1)
                            -> Table scan on <temporary>  (actual time=0.17..0.2 rows=37 loops=1)
                                -> Aggregate using temporary table  (actual time=0.169..0.169 rows=37 loops=1)
                                    -> Table scan on possible_diseases  (cost=4.75 rows=45) (actual time=0.0556..0.0789 rows=45 loops=1)
                            -> Select #3 (subquery in condition; run only once)
                                -> Aggregate: max(sub.incount)  (cost=2.5..2.5 rows=1) (actual time=0.125..0.125 rows=1 loops=1)
                                    -> Table scan on sub  (cost=2.5..2.5 rows=0) (actual time=0.116..0.119 rows=37 loops=1)
                                        -> Materialize  (cost=0..0 rows=0) (actual time=0.115..0.115 rows=37 loops=1)
                                            -> Table scan on <temporary>  (actual time=0.102..0.106 rows=37 loops=1)
                                                -> Aggregate using temporary table  (actual time=0.102..0.102 rows=37 loops=1)
                                                    -> Table scan on possible_diseases  (cost=4.75 rows=45) (actual time=0.0024..0.0778 rows=45 loops=1)
            -> Single-row index lookup on diseases using PRIMARY (DiseaseName=countdis.DiseaseName)  (cost=0.252 rows=1) (actual time=0.00274..0.00276 rows=1 loops=8)
 |
+------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.0028 sec)
```

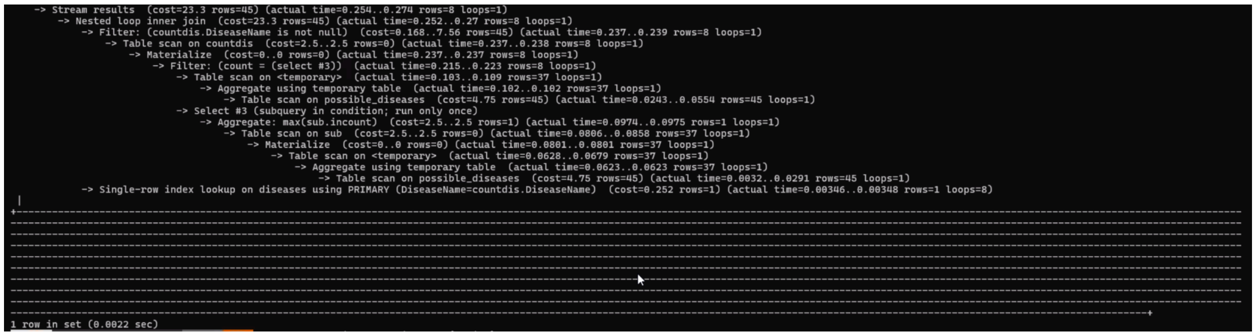
Indexing possible_diseases(diseaseName) -



```
| -> Sort: diseases.Rarity DESC  (actual time=0.176..0.177 rows=8 loops=1)
    -> Stream results  (cost=23.3 rows=45) (actual time=0.151..0.164 rows=8 loops=1)
        -> Nested loop inner join  (cost=23.3 rows=45) (actual time=0.149..0.161 rows=8 loops=1)
            -> Filter: (countdis.DiseaseName is not null)  (cost=12.8..7.56 rows=45) (actual time=0.138..0.139 rows=8 loops=1)
                -> Table scan on countdis  (cost=13..15.9 rows=37) (actual time=0.137..0.138 rows=8 loops=1)
                    -> Materialize  (cost=12.9..12.9 rows=37) (actual time=0.137..0.137 rows=8 loops=1)
                        -> Filter: (count = (select #3))  (cost=9.25 rows=37) (actual time=0.0954..0.126 rows=8 loops=1)
                            -> Group aggregate: count(possible_diseases.DiseaseName)  (cost=9.25 rows=37) (actual time=0.0309..0.059 rows=37 loops=1)
                                -> Covering index scan on possible_diseases using idx_1  (cost=4.75 rows=45) (actual time=0.0209..0.044 rows=45 loops=1)
                            -> Select #3 (subquery in condition; run only once)
                                -> Aggregate: max(sub.incount)  (cost=19.6..19.6 rows=1) (actual time=0.0577..0.0577 rows=1 loops=1)
                                    -> Table scan on sub  (cost=13..15.9 rows=37) (actual time=0.0489..0.0531 rows=37 loops=1)
                                        -> Materialize  (cost=12.9..12.9 rows=37) (actual time=0.048..0.048 rows=37 loops=1)
                                            -> Group aggregate: count(possible_diseases.DiseaseName)  (cost=9.25 rows=37) (actual time=0.0042..0.0338 rows=37 loops=1)
                                                -> Covering index scan on possible_diseases using idx_1  (cost=4.75 rows=45) (actual time=0.0022..0.0232 rows=45 loops=1)
            -> Single-row index lookup on diseases using PRIMARY (DiseaseName=countdis.DiseaseName)  (cost=0.252 rows=1) (actual time=0.00244..0.00249 rows=1 loops=8)
 |
+------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.0016 sec)
```

This gives us a loss in performance, since it increases the cost of aggregation of the DiseaseName in possible_diseases.

Indexing Disease(rarity) -



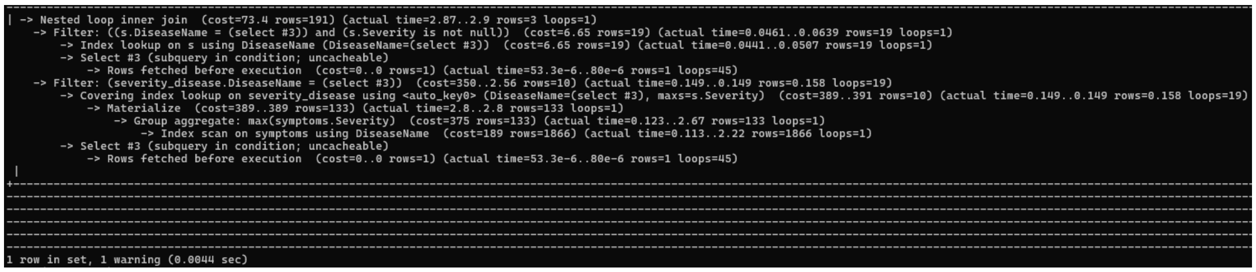
```
    -> Stream results  (cost=23.3 rows=45) (actual time=0.254..0.274 rows=8 loops=1)
        -> Nested loop inner join  (cost=23.3 rows=45) (actual time=0.252..0.27 rows=8 loops=1)
            -> Filter: (countdis.DiseaseName is not null)  (cost=0.168..7.56 rows=45) (actual time=0.237..0.239 rows=8 loops=1)
                -> Table scan on countdis  (cost=2.5..2.5 rows=0) (actual time=0.237..0.238 rows=8 loops=1)
                    -> Materialize  (cost=0..0 rows=0) (actual time=0.237..0.237 rows=8 loops=1)
                        -> Filter: (count = (select #3))  (actual time=0.215..0.223 rows=8 loops=1)
                            -> Table scan on <temporary>  (actual time=0.103..0.109 rows=37 loops=1)
                                -> Aggregate using temporary table  (actual time=0.102..0.102 rows=37 loops=1)
                                    -> Table scan on possible_diseases  (cost=4.75 rows=45) (actual time=0.0243..0.0554 rows=45 loops=1)
                            -> Select #3 (subquery in condition; run only once)
                                -> Aggregate: max(sub.incount)  (cost=2.5..2.5 rows=1) (actual time=0.0974..0.0975 rows=1 loops=1)
                                    -> Table scan on sub  (cost=2.5..2.5 rows=0) (actual time=0.0806..0.0858 rows=37 loops=1)
                                        -> Materialize  (cost=0..0 rows=0) (actual time=0.0801..0.0801 rows=37 loops=1)
                                            -> Table scan on <temporary>  (actual time=0.0628..0.0679 rows=37 loops=1)
                                                -> Aggregate using temporary table  (actual time=0.0623..0.0623 rows=37 loops=1)
                                                    -> Table scan on possible_diseases  (cost=4.75 rows=45) (actual time=0.0032..0.0291 rows=45 loops=1)
            -> Single-row index lookup on diseases using PRIMARY (DiseaseName=countdis.DiseaseName)  (cost=0.252 rows=1) (actual time=0.00346..0.00348 rows=1 loops=8)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.0022 sec)
```

This does not cause any change in performance since rarity is only used to order the final list of likely diseases, not select them so indexing it does not affect performance.

Final Index Design: The final index design we selected was the default indexing as the only relevant attribute for this query is Diseases(DiseaseName) which is a Primary Key and so is already indexed.

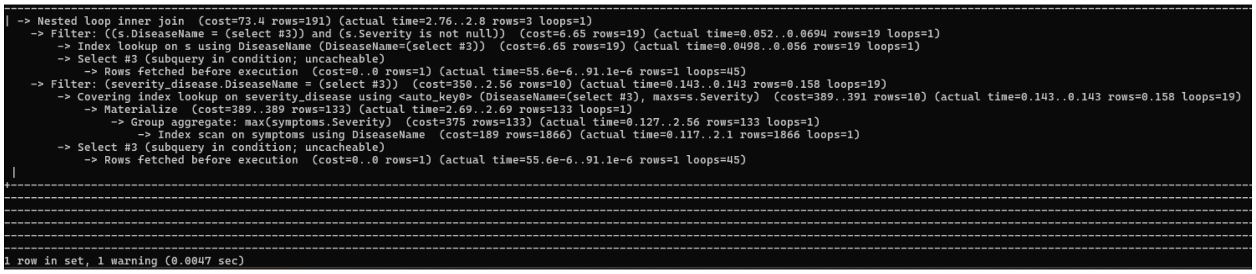
- Query 3 :

Default-



```
-------------------------------------------------------------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=73.4 rows=191) (actual time=2.87..2.9 rows=3 loops=1)
    -> Filter: ((s.DiseaseName = (select #3)) and (s.Severity is not null))  (cost=6.65 rows=19) (actual time=0.0461..0.0639 rows=19 loops=1)
        -> Index lookup on s using DiseaseName (DiseaseName=(select #3))  (cost=6.65 rows=19) (actual time=0.0441..0.0507 rows=19 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=53.3e-6..80e-6 rows=1 loops=45)
    -> Filter: (severity_disease.DiseaseName = (select #3))  (cost=350..2.56 rows=10) (actual time=0.149..0.149 rows=0.158 loops=19)
        -> Covering index lookup on severity_disease using <auto_key0> (DiseaseName=(select #3), maxs=s.Severity)  (cost=389..391 rows=10) (actual time=0.149..0.149 rows=0.158 loops=19)
            -> Materialize  (cost=389..389 rows=133) (actual time=2.8..2.8 rows=133 loops=1)
                -> Group aggregate: max(symptoms.Severity)  (cost=375 rows=133) (actual time=0.123..2.67 rows=133 loops=1)
                    -> Index scan on symptoms using DiseaseName  (cost=189 rows=1866) (actual time=0.113..2.22 rows=1866 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=53.3e-6..80e-6 rows=1 loops=45)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set, 1 warning (0.0044 sec)
```

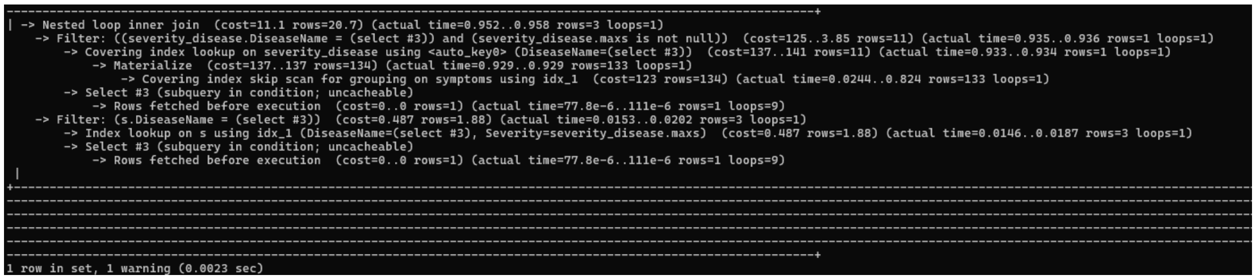
Indexing Symptoms(SymptomName) output-



```
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=73.4 rows=191) (actual time=2.76..2.8 rows=3 loops=1)
    -> Filter: ((s.DiseaseName = (select #3)) and (s.Severity is not null))  (cost=6.65 rows=19) (actual time=0.052..0.0694 rows=19 loops=1)
        -> Index lookup on s using DiseaseName (DiseaseName=(select #3))  (cost=6.65 rows=19) (actual time=0.0498..0.056 rows=19 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=55.6e-6..91.1e-6 rows=1 loops=45)
    -> Filter: (severity_disease.DiseaseName = (select #3))  (cost=350..2.56 rows=10) (actual time=0.143..0.143 rows=0.158 loops=19)
        -> Covering index lookup on severity_disease using <auto_key0> (DiseaseName=(select #3), maxs=s.Severity)  (cost=389..391 rows=10) (actual time=0.143..0.143 rows=0.158 loops=19)
            -> Materialize  (cost=389..389 rows=133) (actual time=2.69..2.69 rows=133 loops=1)
                -> Group aggregate: max(symptoms.Severity)  (cost=375 rows=133) (actual time=0.127..2.56 rows=133 loops=1)
                    -> Index scan on symptoms using DiseaseName  (cost=189 rows=1866) (actual time=0.117..2.1 rows=1866 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=55.6e-6..91.1e-6 rows=1 loops=45)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set, 1 warning (0.0047 sec)
```

This does not change the performance of the query as we can see that the cost at every step remains the same. This is because our query only displays SymptomName and does not use it in any computation.

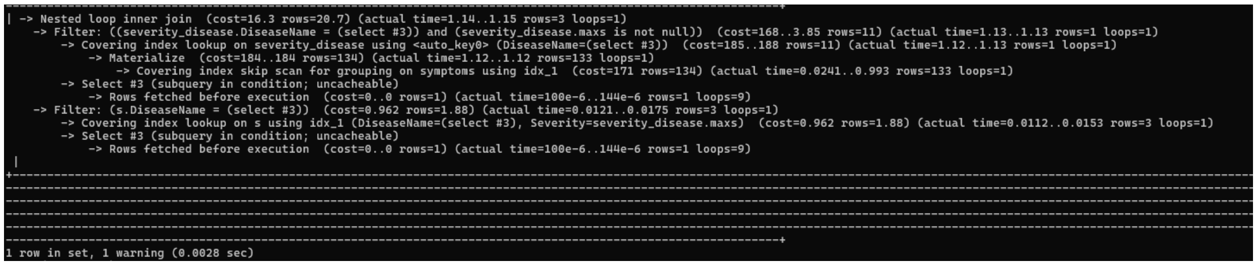
Indexing Symptoms(DiseaseName, Severity)



```
--------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=11.1 rows=20.7) (actual time=0.952..0.958 rows=3 loops=1)
    -> Filter: ((severity_disease.DiseaseName = (select #3)) and (severity_disease.maxs is not null))  (cost=125..3.85 rows=11) (actual time=0.935..0.936 rows=1 loops=1)
        -> Covering index lookup on severity_disease using <auto_key0> (DiseaseName=(select #3))  (cost=137..141 rows=11) (actual time=0.933..0.934 rows=1 loops=1)
            -> Materialize  (cost=137..137 rows=134) (actual time=0.929..0.929 rows=133 loops=1)
                -> Covering index skip scan for grouping on symptoms using idx_1  (cost=123 rows=134) (actual time=0.0244..0.824 rows=133 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=77.8e-6..111e-6 rows=1 loops=9)
    -> Filter: (s.DiseaseName = (select #3))  (cost=0.487 rows=1.88) (actual time=0.0153..0.0202 rows=3 loops=1)
        -> Index lookup on s using idx_1 (DiseaseName=(select #3), Severity=severity_disease.maxs)  (cost=0.487 rows=1.88) (actual time=0.0146..0.0187 rows=3 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=77.8e-6..111e-6 rows=1 loops=9)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------+
1 row in set, 1 warning (0.0023 sec)
```

This gave us a great performance gain as the nested loop which was joined with the Symptoms table (predicted_disease) could be computed with less cost as well as the join itself since this indexing allows for faster data access.

Indexing Symptoms(DiseaseName, Severity, SymptomName)



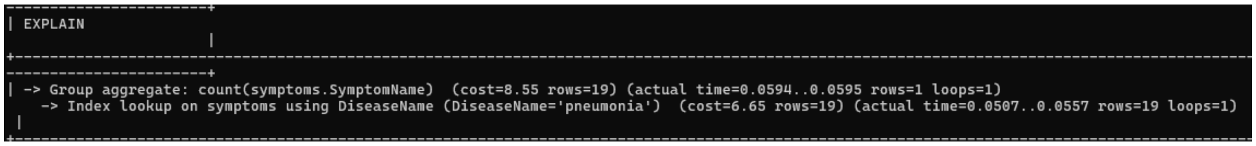
```
---------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=16.3 rows=20.7) (actual time=1.14..1.15 rows=3 loops=1)
    -> Filter: ((severity_disease.DiseaseName = (select #3)) and (severity_disease.maxs is not null))  (cost=168..3.85 rows=11) (actual time=1.13..1.13 rows=1 loops=1)
        -> Covering index lookup on severity_disease using <auto_key0> (DiseaseName=(select #3))  (cost=185..188 rows=11) (actual time=1.12..1.13 rows=1 loops=1)
            -> Materialize  (cost=184..184 rows=134) (actual time=1.12..1.12 rows=133 loops=1)
                -> Covering index skip scan for grouping on symptoms using idx_1  (cost=171 rows=134) (actual time=0.0241..0.993 rows=133 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=100e-6..144e-6 rows=1 loops=9)
    -> Filter: (s.DiseaseName = (select #3))  (cost=0.962 rows=1.88) (actual time=0.0121..0.0175 rows=3 loops=1)
        -> Covering index lookup on s using idx_1 (DiseaseName=(select #3), Severity=severity_disease.maxs)  (cost=0.962 rows=1.88) (actual time=0.0112..0.0153 rows=3 loops=1)
        -> Select #3 (subquery in condition; uncacheable)
            -> Rows fetched before execution  (cost=0..0 rows=1) (actual time=100e-6..144e-6 rows=1 loops=9)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------+
1 row in set, 1 warning (0.0028 sec)
```

This indexing does increase performance when compared to the default query but when compared to indexing Symptoms(DiseaseName, Severity) the performance decreases. This is because indexing SymptomName as well increases the cost of the join.

<u>Final Index Design</u>: The final index design we selected was indexing Symptoms(DiseaseName, Severity) since it gave us the greatest performance gain (cost of the nested loop and the join decreased).

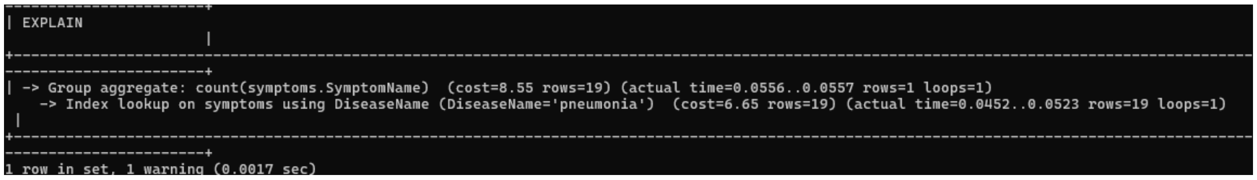
- Query 4 :

Default-



```
----------------------+
| EXPLAIN
                      |
+---------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Group aggregate: count(symptoms.SymptomName)  (cost=8.55 rows=19) (actual time=0.0594..0.0595 rows=1 loops=1)
    -> Index lookup on symptoms using DiseaseName (DiseaseName='pneumonia')  (cost=6.65 rows=19) (actual time=0.0507..0.0557 rows=19 loops=1)
 |
+---------------------------------------------------------------------------------------------------------------------------------
```

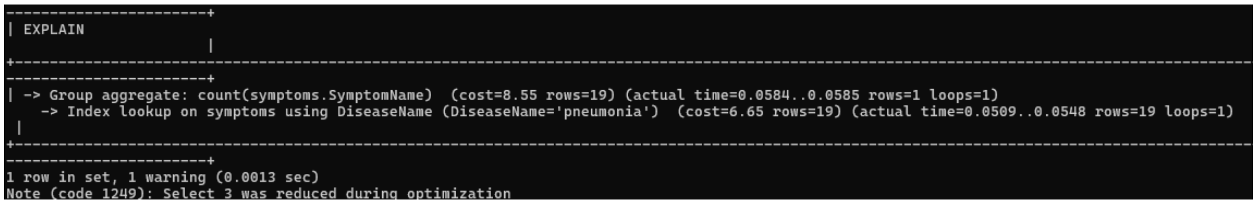
Indexing Symptoms(SymptomName)-



```
----------------------+
| EXPLAIN
                      |
+---------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Group aggregate: count(symptoms.SymptomName)  (cost=8.55 rows=19) (actual time=0.0556..0.0557 rows=1 loops=1)
    -> Index lookup on symptoms using DiseaseName (DiseaseName='pneumonia')  (cost=6.65 rows=19) (actual time=0.0452..0.0523 rows=19 loops=1)
 |
+---------------------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set, 1 warning (0.0017 sec)
```

This does not change the performance of the query as we can see that the cost at every step remains the same. This is because our query only displays the aggregate of SymptomName so the indexing doesn't matter.

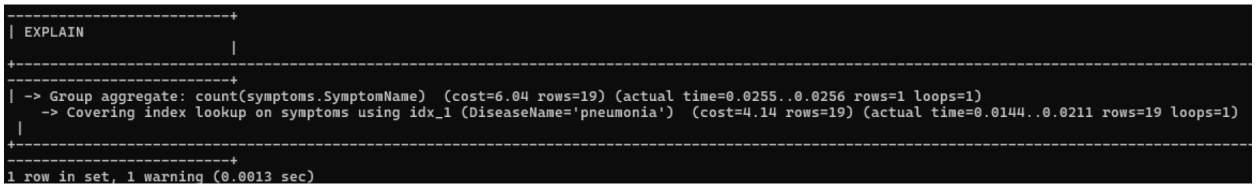
Indexing Symptoms(SymptomName, DiseaseName)-



```
----------------------+
| EXPLAIN
                      |
+---------------------------------------------------------------------------------------------------------------------------------
----------------------+
| -> Group aggregate: count(symptoms.SymptomName)  (cost=8.55 rows=19) (actual time=0.0584..0.0585 rows=1 loops=1)
    -> Index lookup on symptoms using DiseaseName (DiseaseName='pneumonia')  (cost=6.65 rows=19) (actual time=0.0509..0.0548 rows=19 loops=1)
 |
+---------------------------------------------------------------------------------------------------------------------------------
----------------------+
1 row in set, 1 warning (0.0013 sec)
Note (code 1249): Select 3 was reduced during optimization
```

The indexing does not change performance due to the same reason as above. Since our query's join and group by condition is focused only on DiseaseName, the performance is the same.

Indexing Symptoms(DiseaseName, SymptomName)-



```
-------------------------+
| EXPLAIN
                         |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| -> Group aggregate: count(symptoms.SymptomName)  (cost=6.04 rows=19) (actual time=0.0255..0.0256 rows=1 loops=1)
    -> Covering index lookup on symptoms using idx_1 (DiseaseName='pneumonia')  (cost=4.14 rows=19) (actual time=0.0144..0.0211 rows=19 loops=1)
 |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------+
1 row in set, 1 warning (0.0013 sec)
```

This indexing improves the performance of our query. This is because the cost of grouping is decreased and this indexing allows for faster data access.

<u>Final Index Design</u>: The final index design we selected was indexing Symptoms(DiseaseName, SymptomName) as it improved the efficiency of our query the most. The cost of grouping was decreased and this indexing allows for the fastest data access.