Team Members: Riley Morris, Dimple Patel, Prathmesh Rathod

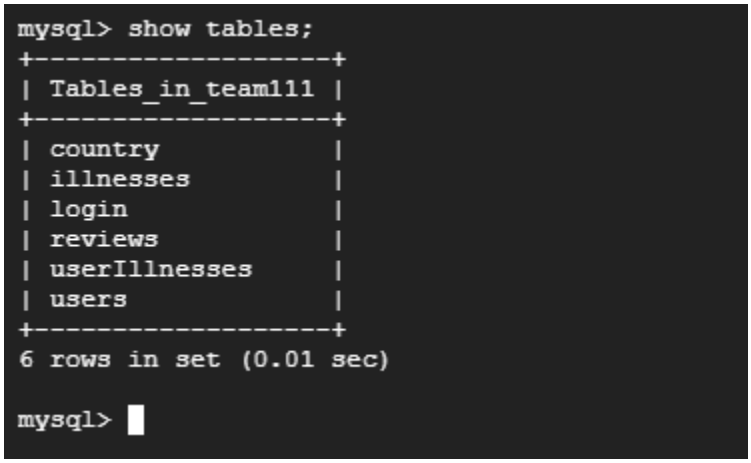
Team Name: Health Guide Team-111

Project 1 Stage 3

Six Main Tables:

- Users
- Illnesses
- Country
- Review
- Login
- UserIllnesses

Proof of database and table connection:



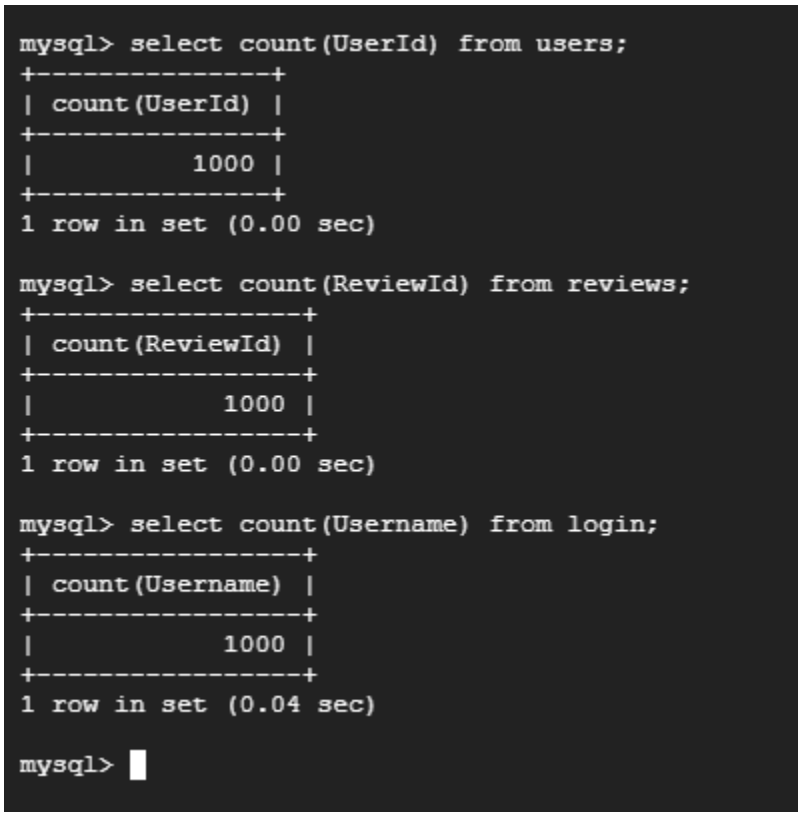
DDL Commands to create tables (on next page):

```sql
CREATE DATABASE /*!32312 IF NOT EXISTS*/`team111` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `team111`;

DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
  `UserId` int(11) NOT NULL,
  `Age` int(11) NOT NULL,
  `Gender` varchar(50) NOT NULL,
  `Country` varchar(50) NOT NULL,
  PRIMARY KEY (`UserId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `country`;
CREATE TABLE `country` (
  `CountryId` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Population` int(11) NOT NULL,
  `CommonIllnesses` varchar(1023) NOT NULL,
  PRIMARY KEY (`CountryId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `reviews`;
CREATE TABLE `reviews` (
  `ReviewId` int(11) NOT NULL,
  `UserId` int(11) NOT NULL,
  `Title` varchar(255) NOT NULL,
  `IllnessName` varchar(255) NOT NULL,
  `ReviewBody` varchar(4095) NOT NULL,
  PRIMARY KEY (`ReviewId`),
  KEY `UserId` (`UserId`),
  CONSTRAINT `reviews_ibfk_1` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `illnesses`;
CREATE TABLE `illnesses` (
  `IllnessId` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Symptoms` varchar(255) NOT NULL,
  `Treatments` varchar(255) NOT NULL,
  PRIMARY KEY (`IllnessId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `login`;
CREATE TABLE `login` (
  `Username` varchar(255) NOT NULL,
  `UserId` int(11) NOT NULL,
  `Password` varchar(255) NOT NULL,
  PRIMARY KEY (`Username`),
  KEY `UserId` (`UserId`),
  CONSTRAINT `login_ibfk_1` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

DROP TABLE IF EXISTS `userIllnesses`;
CREATE TABLE `userIllnesses` (
  `UserId` int(11) NOT NULL,
  `IllnessId` int(11) NOT NULL,
  PRIMARY KEY (`UserId`, `IllnessId`),
  KEY `IllnessId` (`IllnessId`),
  CONSTRAINT `userIllnesses_ibfk_1` FOREIGN KEY (`UserId`) REFERENCES `users` (`UserId`),
  CONSTRAINT `userIllnesses_ibfk_2` FOREIGN KEY (`IllnessId`) REFERENCES `ilnesses` (`IllnessId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Tables with 1000+ entries:

- Users
- Login
- Reviews

Proof:



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

mysql> select count(ReviewId) from reviews;
+-----------------+
| count(ReviewId) |
+-----------------+
|            1000 |
+-----------------+
1 row in set (0.00 sec)

mysql> select count(Username) from login;
+-----------------+
| count(Username) |
+-----------------+
|            1000 |
+-----------------+
1 row in set (0.04 sec)

mysql>
```

Advanced Query 1:

```
SELECT IllnessName, AVG(Age)
FROM users u JOIN reviews r using(UserId)
GROUP BY IllnessName;
```

First 15 rows:

```
mysql> SELECT IllnessName, AVG(Age)
    -> FROM users u JOIN reviews r using(UserId)
    -> GROUP BY IllnessName;
+-------------------------+----------+
| IllnessName             | AVG(Age) |
+-------------------------+----------+
| Gastroparesis           |  42.2000 |
| Chrohn's disease        |  42.4235 |
| Heart palpitations      |  37.9808 |
| Anemia                  |  43.7571 |
| Irritable bowel syndrome |  39.4571 |
| Strep Throat            |  40.6364 |
| Depression              |  41.9048 |
| Anxiety                 |  41.4928 |
| Fybromyalgia            |  37.4615 |
| Elhers-Danlos syndrome  |  42.5893 |
| Anhedonia               |  43.7000 |
| Sinus headaches         |  42.3750 |
| TMJ disorders           |  38.6724 |
| Ulcerative colitis      |  43.6984 |
| Influenza               |  45.4348 |
+-------------------------+----------+
15 rows in set (0.01 sec)

mysql>
```

Explanation:

The query returns the average age of all users who suffered from each disease. This can be done by joining our user and review tables, and then grouping by the name of each illness. This information is important because we will want to give average age ranges on our final site so users can know what diseases are typically within their age range. As for the actual results, we gave the users randomly generated ages from 1 to 80, so the actual results of the query are not the most useful yet.

Advanced Query 2:

```
SELECT IllnessName, count(u.UserId)
FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name =
r.IllnessName) JOIN country c on (u.Country = c.Name)
WHERE (i.Name = "Influenza") AND (u.age >= 20) AND (u.Gender = "Female")
GROUP BY IllnessName

UNION

SELECT IllnessName, count(u.UserId)
FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name =
r.IllnessName) JOIN country c on (u.Country = c.Name)
WHERE (i.Name = "Strep Throat") AND (u.age >= 20) AND (u.Gender = "Female")
GROUP BY IllnessName;
```

First 2 rows (we did not have 15 rows):

```
mysql> SELECT IllnessName, count(u.UserId)
    -> FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name = r.IllnessName) JOIN country c on (u.Country = c.Name)
    -> WHERE (i.Name = "Influenza") AND (u.age >= 20) AND (u.Gender = "Female")
    -> GROUP BY IllnessName
    ->
    -> UNION
    ->
    -> SELECT IllnessName, count(u.UserId)
    -> FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name = r.IllnessName) JOIN country c on (u.Country = c.Name)
    -> WHERE (i.Name = "Strep Throat") AND (u.age >= 20) AND (u.Gender = "Female")
    -> GROUP BY IllnessName;
+--------------+-----------------+
| IllnessName  | count(u.UserId) |
+--------------+-----------------+
| Influenza    |              31 |
| Strep Throat |              19 |
+--------------+-----------------+
2 rows in set (0.01 sec)

mysql>
```

Explanation:

The query returns the number of females above the age of 20 who have either had Influenza or Strep Throat. The idea of this query isn't this exact query. The goal is to have users of our application enter in some set of constraints (such as the ones in this query) so they can get a better understanding of how common their illness is. For example, if a user wants to see the number of men under the age of 20 who develop Crohn's Disease, they will see that this is unlikely given that Crohn's disease may generally be developed later in life. These queries will better help users understand these illnesses.

Advanced Query 1 Explain Analyze:

SELECT IllnessName, AVG(Age)
FROM users u JOIN reviews r using(UserId)
GROUP BY IllnessName;

To tackle this query we tried three different indexes. The first index we tried was UserId, but this made no changes as it was the primary key, and users were already sorted by this key. We had to try it to get three different indexes. The second index we tried was more helpful. We created an index for the reviews table on IllnessName. It did not lower the cost, however, it lowered the actual time of the table scan and nested loop inner join. Finally, we added an index for Age on the users table. Again, this lowered the actual time of the query, but not the cost.

Original Results:

```
-> Table scan on <temporary>  (actual time=0.000..0.002 rows=15 loops=1)
  -> Aggregate using temporary table  (actual time=2.398..2.400 rows=15 loops=1)
      -> Nested loop inner join  (cost=453.50 rows=1000) (actual time=0.272..1.675 rows=1000 loops=1)
          -> Table scan on r  (cost=103.50 rows=1000) (actual time=0.232..0.544 rows=1000 loops=1)
          -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1000)
```

After IllnessName index:

```
mysql> CREATE INDEX rIllnessName on reviews(IllnessName);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT IllnessName, AVG(Age) FROM users u JOIN reviews r using(UserId) GROUP BY IllnessName;
| EXPLAIN
| -> Table scan on <temporary>  (actual time=0.001..0.002 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=2.245..2.247 rows=15 loops=1)
        -> Nested loop inner join  (cost=453.50 rows=1000) (actual time=0.066..1.463 rows=1000 loops=1)
            -> Table scan on r  (cost=103.50 rows=1000) (actual time=0.055..0.367 rows=1000 loops=1)
            -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1000)
```

After Age index:

```
mysql> CREATE INDEX uAge on users(Age);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT IllnessName, AVG(Age) FROM users u JOIN reviews r using(UserId) GROUP BY IllnessName;
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------+
| EXPLAIN

                       |
+-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
-----------------------+
| -> Table scan on <temporary>  (actual time=0.001..0.002 rows=15 loops=1)
    -> Aggregate using temporary table  (actual time=2.148..2.151 rows=15 loops=1)
        -> Nested loop inner join  (cost=453.50 rows=1000) (actual time=0.047..1.445 rows=1000 loops=1)
            -> Table scan on r  (cost=103.50 rows=1000) (actual time=0.038..0.383 rows=1000 loops=1)
            -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1000)
 |
```

Explanation:

- We were unable to lower the cost of the query because the main cost is derived from the table join, and the table join is using UserId, a primary key that is already sorted
- Adding in indexes for IllnessName and Age decreased the actual time by a good amount, but did not lower cost since they were only used to display at the end of the query

Advanced Query 2 Explain Analyze:

```
SELECT IllnessName, count(u.UserId)
FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name =
r.IllnessName) JOIN country c on (u.Country = c.Name)
WHERE (i.Name = "Influenza") AND (u.age >= 20) AND (u.Gender = "Female")
GROUP BY IllnessName

UNION

SELECT IllnessName, count(u.UserId)
FROM users u JOIN reviews r ON (u.userId = r.userId) JOIN illnesses i on (i.Name =
r.IllnessName) JOIN country c on (u.Country = c.Name)
WHERE (i.Name = "Strep Throat") AND (u.age >= 20) AND (u.Gender = "Female")
GROUP BY IllnessName;
```

We tried numerous different indexes on this query. We tried to index the IllnessName in the reviews table, the country index on the users table, and the gender index on the users table. In all three cases none of these changed the cost. We think that this may be due to the sorting of these items having little impact on the joins since the joined tables are already sorted on these keys.

Original Results:

```
| -> Table scan on <union temporary>  (cost=0.08..2.95 rows=36) (actual time=0.001..0.001 rows=2 loops=1)
    -> Union materialize with deduplication  (cost=126.66..129.53 rows=36) (actual time=0.995..0.995 rows=2 loops=1)
        -> Group aggregate: count(u.UserId)  (cost=66.96 rows=20) (actual time=0.504..0.504 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=64.94 rows=20) (actual time=0.416..0.498 rows=31 loops=1)
                -> Table scan on c  (cost=0.40 rows=248) (actual time=0.023..0.066 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=43.98 rows=1) (actual time=0.350..0.355 rows=31 loops=1)
                        -> Filter: (i.`Name` = 'Influenza')  (cost=0.32 rows=2) (actual time=0.014..0.016 rows=1 loops=1)
                            -> Table scan on i  (cost=0.32 rows=15) (actual time=0.011..0.013 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=41.55 rows=5) (actual time=0.181..0.310 rows=31 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Influenza')  (cost=17.40 rows=69) (actual time=0.138..0.149 rows=69 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=69)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=69)
        -> Group aggregate: count(u.UserId)  (cost=55.98 rows=16) (actual time=0.474..0.475 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=54.37 rows=16) (actual time=0.400..0.472 rows=19 loops=1)
                -> Table scan on c  (cost=0.50 rows=248) (actual time=0.021..0.059 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=37.51 rows=1) (actual time=0.323..0.327 rows=19 loops=1)
                        -> Filter: (i.`Name` = 'Strep Throat')  (cost=0.41 rows=2) (actual time=0.010..0.012 rows=1 loops=1)
                            -> Table scan on i  (cost=0.41 rows=15) (actual time=0.008..0.010 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=35.25 rows=4) (actual time=0.184..0.296 rows=19 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Strep Throat')  (cost=16.00 rows=55) (actual time=0.173..0.184 rows=55 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=55)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=55)
|
```

After IllnessName on reviews table:

```
| -> Table scan on <union temporary>  (cost=0.08..2.95 rows=36) (actual time=0.001..0.001 rows=2 loops=1)
    -> Union materialize with deduplication  (cost=126.66..129.53 rows=36) (actual time=0.903..0.903 rows=2 loops=1)
        -> Group aggregate: count(u.UserId)  (cost=66.96 rows=20) (actual time=0.508..0.508 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=64.94 rows=20) (actual time=0.425..0.504 rows=31 loops=1)
                -> Table scan on c  (cost=0.40 rows=248) (actual time=0.024..0.063 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=43.98 rows=1) (actual time=0.328..0.333 rows=31 loops=1)
                        -> Filter: (i.`Name` = 'Influenza')  (cost=0.32 rows=2) (actual time=0.013..0.015 rows=1 loops=1)
                            -> Table scan on i  (cost=0.32 rows=15) (actual time=0.010..0.013 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=41.55 rows=5) (actual time=0.173..0.294 rows=31 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Influenza')  (cost=17.40 rows=69) (actual time=0.155..0.164 rows=69 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=69)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=69)
        -> Group aggregate: count(u.UserId)  (cost=55.98 rows=16) (actual time=0.379..0.379 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=54.37 rows=16) (actual time=0.305..0.377 rows=19 loops=1)
                -> Table scan on c  (cost=0.50 rows=248) (actual time=0.021..0.058 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=37.51 rows=1) (actual time=0.254..0.258 rows=19 loops=1)
                        -> Filter: (i.`Name` = 'Strep Throat')  (cost=0.41 rows=2) (actual time=0.009..0.010 rows=1 loops=1)
                            -> Table scan on i  (cost=0.41 rows=15) (actual time=0.006..0.008 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=35.25 rows=4) (actual time=0.148..0.232 rows=19 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Strep Throat')  (cost=16.00 rows=55) (actual time=0.135..0.142 rows=55 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=55)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=55)
 |
```

After Country index:

```
| -> Table scan on <union temporary>  (cost=0.08..2.95 rows=36) (actual time=0.001..0.001 rows=2 loops=1)
    -> Union materialize with deduplication  (cost=126.66..129.53 rows=36) (actual time=0.852..0.852 rows=2 loops=1)
        -> Group aggregate: count(u.UserId)  (cost=66.96 rows=20) (actual time=0.480..0.480 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=64.94 rows=20) (actual time=0.392..0.475 rows=31 loops=1)
                -> Table scan on c  (cost=0.40 rows=248) (actual time=0.022..0.062 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=43.98 rows=1) (actual time=0.304..0.309 rows=31 loops=1)
                        -> Filter: (i.`Name` = 'Influenza')  (cost=0.32 rows=2) (actual time=0.019..0.021 rows=1 loops=1)
                            -> Table scan on i  (cost=0.32 rows=15) (actual time=0.016..0.019 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=41.55 rows=5) (actual time=0.147..0.262 rows=31 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Influenza')  (cost=17.40 rows=69) (actual time=0.129..0.138 rows=69 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=69)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=69)
        -> Group aggregate: count(u.UserId)  (cost=55.98 rows=16) (actual time=0.358..0.358 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=54.37 rows=16) (actual time=0.239..0.355 rows=19 loops=1)
                -> Table scan on c  (cost=0.50 rows=248) (actual time=0.020..0.071 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=37.51 rows=1) (actual time=0.192..0.196 rows=19 loops=1)
                        -> Filter: (i.`Name` = 'Strep Throat')  (cost=0.41 rows=2) (actual time=0.007..0.009 rows=1 loops=1)
                            -> Table scan on i  (cost=0.41 rows=15) (actual time=0.005..0.007 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=35.25 rows=4) (actual time=0.095..0.171 rows=19 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Strep Throat')  (cost=16.00 rows=55) (actual time=0.084..0.091 rows=55 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=55)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=55)
 |
```

After using the Country index, there was no significant change in the cost and actual time.

After Gender index:

```
| -> Table scan on <union temporary>  (cost=0.03..4.38 rows=150) (actual time=0.001..0.001 rows=2 loops=1)
    -> Union materialize with deduplication  (cost=1073.90..1078.25 rows=150) (actual time=5.081..5.081 rows=2 loops=1)
        -> Group aggregate: count(u.UserId)  (cost=927.50 rows=68) (actual time=4.718..4.718 rows=1 loops=1)
            -> Nested loop inner join  (cost=920.70 rows=68) (actual time=0.426..4.708 rows=31 loops=1)
                -> Nested loop inner join  (cost=575.50 rows=614) (actual time=0.085..3.313 rows=403 loops=1)
                    -> Inner hash join (no condition)  (cost=39.70 rows=372) (actual time=0.064..0.178 rows=248 loops=1)
                        -> Table scan on c  (cost=17.03 rows=248) (actual time=0.024..0.084 rows=248 loops=1)
                        -> Hash
                            -> Filter: (i.`Name` = 'Influenza')  (cost=1.75 rows=2) (actual time=0.025..0.027 rows=1 loops=1)
                                -> Table scan on i  (cost=1.75 rows=15) (actual time=0.019..0.023 rows=15 loops=1)
                    -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=1.03 rows=2) (actual time=0.010..0.012 rows=2 loops=248)
                        -> Index lookup on u using uCountry (Country=c.`Name`), with index condition: (u.Country = c.`Name`)  (cost=1.03 rows=4) (actual time=0.010..0.012 rows=4 loops=248)
                -> Filter: (r.IllnessName = 'Influenza')  (cost=0.40 rows=0) (actual time=0.003..0.003 rows=0 loops=403)
                    -> Index lookup on r using UserId (UserId=u.UserId)  (cost=0.40 rows=2) (actual time=0.003..0.003 rows=1 loops=403)
        -> Group aggregate: count(u.UserId)  (cost=131.35 rows=82) (actual time=0.345..0.345 rows=1 loops=1)
            -> Inner hash join (u.Country = c.`Name`)  (cost=123.14 rows=82) (actual time=0.260..0.342 rows=19 loops=1)
                -> Table scan on c  (cost=0.10 rows=248) (actual time=0.029..0.069 rows=248 loops=1)
                -> Hash
                    -> Inner hash join (no condition)  (cost=40.21 rows=3) (actual time=0.194..0.198 rows=19 loops=1)
                        -> Filter: (i.`Name` = 'Strep Throat')  (cost=0.08 rows=2) (actual time=0.012..0.013 rows=1 loops=1)
                            -> Table scan on i  (cost=0.08 rows=15) (actual time=0.009..0.012 rows=15 loops=1)
                        -> Hash
                            -> Nested loop inner join  (cost=35.25 rows=22) (actual time=0.090..0.159 rows=19 loops=1)
                                -> Index lookup on r using rIllnessName (IllnessName='Strep Throat')  (cost=16.00 rows=55) (actual time=0.078..0.084 rows=55 loops=1)
                                -> Filter: ((u.Gender = 'Female') and (u.Age >= 20))  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=55)
                                    -> Single-row index lookup on u using PRIMARY (UserId=r.UserId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=55)
 |
```

Explanation:

- We were unable to lower the cost of the query on any of our indexes
- This may be due to the nature of the joins we are committing
- A large portion of the cost is derived from the union operation, this could be reduced by combining the queries into one query. We aren't sure if this is possible with the front end, but if it is then the union should be removed