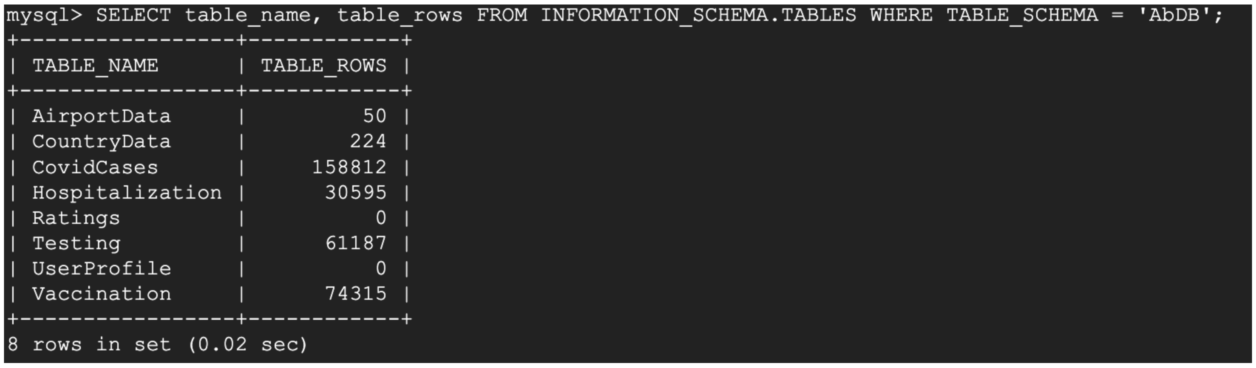
DDL Commands:

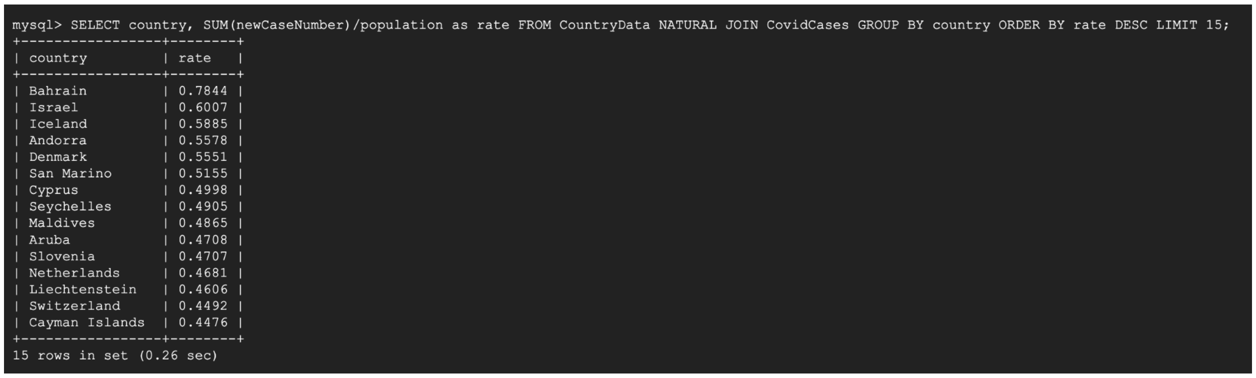
```
CREATE TABLE UserProfile(
userFirstName VARCHAR(100) NOT NULL,
userLastName VARCHAR(100) NOT NULL,
destinationCity VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
password VARCHAR(100) NOT NULL,
PRIMARY KEY(email),
FOREIGN KEY(destinationCity) REFERENCES AirportData(airportCity)
);

CREATE TABLE CountryData(
country VARCHAR(100) NOT NULL,
countryCode VARCHAR(3) NOT NULL,
population INT,
region VARCHAR(100),
PRIMARY KEY(country)
);

CREATE TABLE AirportData(
country VARCHAR(100) NOT NULL,
airportCity VARCHAR(100) NOT NULL,
airportName VARCHAR(100) NOT NULL,
airportCode VARCHAR(3) NOT NULL,
PRIMARY KEY(airportCode),
FOREIGN KEY(country) REFERENCES CountryData(country),
UNIQUE(airportCity),
UNIQUE(airportName),
UNIQUE(airportCode)
);

CREATE TABLE CovidCases(
country VARCHAR(100) NOT NULL,
date TIMESTAMP NOT NULL,
newCaseNumber INT,
newDeathNumber INT,
PRIMARY KEY(date, country),
FOREIGN KEY(country) REFERENCES CountryData(country)
);

CREATE TABLE Vaccination(
country VARCHAR(100) NOT NULL,
date TIMESTAMP NOT NULL,
dailyVaccinationNumber INT,
PRIMARY KEY(date, country),
FOREIGN KEY(country) REFERENCES CountryData(country)
 );

CREATE TABLE Hospitalization(
country VARCHAR(100) NOT NULL,
date TIMESTAMP NOT NULL,
patientNumber INT,
PRIMARY KEY(date, country),
FOREIGN KEY(country) REFERENCES CountryData(country)
);

CREATE TABLE Testing(
country VARCHAR(100) NOT NULL,
date TIMESTAMP NOT NULL,
newTestNumber INT,
PRIMARY KEY(date, country),
FOREIGN KEY(country) REFERENCES CountryData(country)
);

CREATE TABLE Ratings(
airportName VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
rating INT,
review TEXT,
PRIMARY KEY(airportName, email),
FOREIGN KEY(email) REFERENCES UserProfile(email),
FOREIGN KEY(airportName) REFERENCES AirportData(airportName)
);
```

Proof of Database:

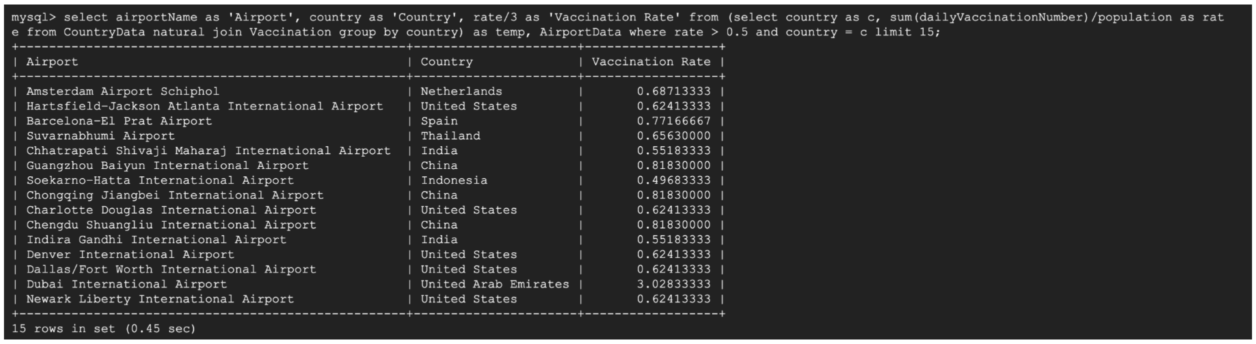


```
mysql> SELECT table_name, table_rows FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'AbDB';
+-----------------+------------+
| TABLE_NAME      | TABLE_ROWS |
+-----------------+------------+
| AirportData     |         50 |
| CountryData     |        224 |
| CovidCases      |     158812 |
| Hospitalization |      30595 |
| Ratings         |          0 |
| Testing         |      61187 |
| UserProfile     |          0 |
| Vaccination     |      74315 |
+-----------------+------------+
8 rows in set (0.02 sec)
```

Advanced SQL Commands:



```
mysql> SELECT country, SUM(newCaseNumber)/population as rate FROM CountryData NATURAL JOIN CovidCases GROUP BY country ORDER BY rate DESC LIMIT 15;
+-----------------+--------+
| country         | rate   |
+-----------------+--------+
| Bahrain         | 0.7844 |
| Israel          | 0.6007 |
| Iceland         | 0.5885 |
| Andorra         | 0.5578 |
| Denmark         | 0.5551 |
| San Marino      | 0.5155 |
| Cyprus          | 0.4998 |
| Seychelles      | 0.4905 |
| Maldives        | 0.4865 |
| Aruba           | 0.4708 |
| Slovenia        | 0.4707 |
| Netherlands     | 0.4681 |
| Liechtenstein   | 0.4606 |
| Switzerland     | 0.4492 |
| Cayman Islands  | 0.4476 |
+-----------------+--------+
15 rows in set (0.26 sec)
```

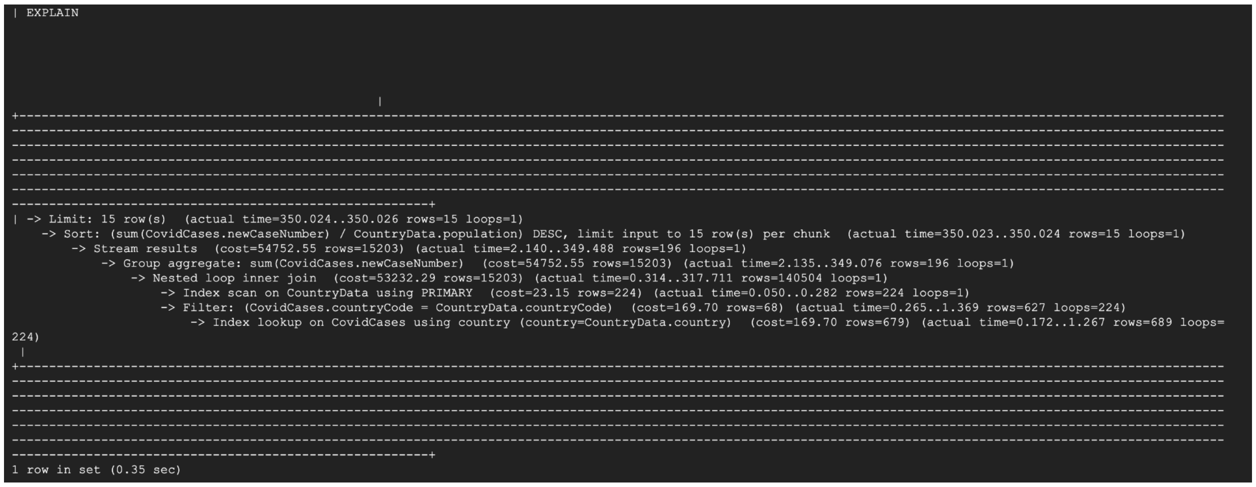


```
mysql> select airportName as 'Airport', country as 'Country', rate/3 as 'Vaccination Rate' from (select country as c, sum(dailyVaccinationNumber)/population as rat
e from CountryData natural join Vaccination group by country) as temp, AirportData where rate > 0.5 and country = c limit 15;
+---------------------------------------------------+----------------------+------------------+
| Airport                                           | Country              | Vaccination Rate |
+---------------------------------------------------+----------------------+------------------+
| Amsterdam Airport Schiphol                        | Netherlands          |       0.68713333 |
| Hartsfield-Jackson Atlanta International Airport  | United States        |       0.62413333 |
| Barcelona-El Prat Airport                         | Spain                |       0.77166667 |
| Suvarnabhumi Airport                              | Thailand             |       0.65630000 |
| Chhatrapati Shivaji Maharaj International Airport | India                |       0.55183333 |
| Guangzhou Baiyun International Airport            | China                |       0.81830000 |
| Soekarno-Hatta International Airport              | Indonesia            |       0.49683333 |
| Chongqing Jiangbei International Airport          | China                |       0.81830000 |
| Charlotte Douglas International Airport           | United States        |       0.62413333 |
| Chengdu Shuangliu International Airport           | China                |       0.81830000 |
| Indira Gandhi International Airport               | India                |       0.55183333 |
| Denver International Airport                      | United States        |       0.62413333 |
| Dallas/Fort Worth International Airport           | United States        |       0.62413333 |
| Dubai International Airport                       | United Arab Emirates |       3.02833333 |
| Newark Liberty International Airport              | United States        |       0.62413333 |
+---------------------------------------------------+----------------------+------------------+
15 rows in set (0.45 sec)
```

Indexing:

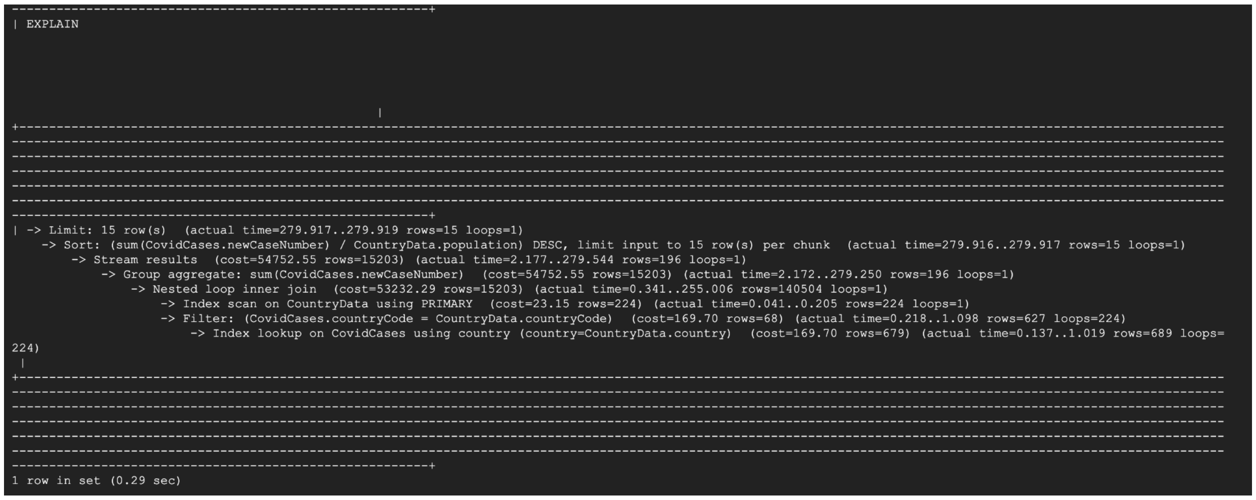
<u>First advanced SQL command:</u>

PRIMARY index:



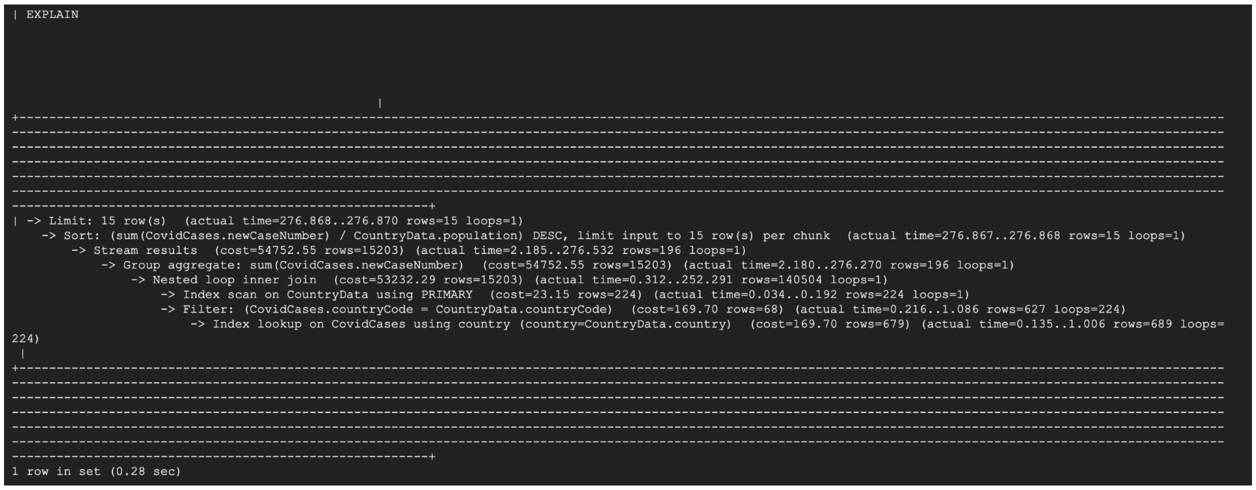
```
| EXPLAIN




                                                  |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=350.024..350.026 rows=15 loops=1)
    -> Sort: (sum(CovidCases.newCaseNumber) / CountryData.population) DESC, limit input to 15 row(s) per chunk  (actual time=350.023..350.024 rows=15 loops=1)
        -> Stream results  (cost=54752.55 rows=15203) (actual time=2.140..349.488 rows=196 loops=1)
            -> Group aggregate: sum(CovidCases.newCaseNumber)  (cost=54752.55 rows=15203) (actual time=2.135..349.076 rows=196 loops=1)
                -> Nested loop inner join  (cost=53232.29 rows=15203) (actual time=0.314..317.711 rows=140504 loops=1)
                    -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.050..0.282 rows=224 loops=1)
                    -> Filter: (CovidCases.countryCode = CountryData.countryCode)  (cost=169.70 rows=68) (actual time=0.265..1.369 rows=627 loops=224)
                        -> Index lookup on CovidCases using country (country=CountryData.country)  (cost=169.70 rows=679) (actual time=0.172..1.267 rows=689 loops=
224)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
1 row in set (0.35 sec)
```

PRIMARY and population index:



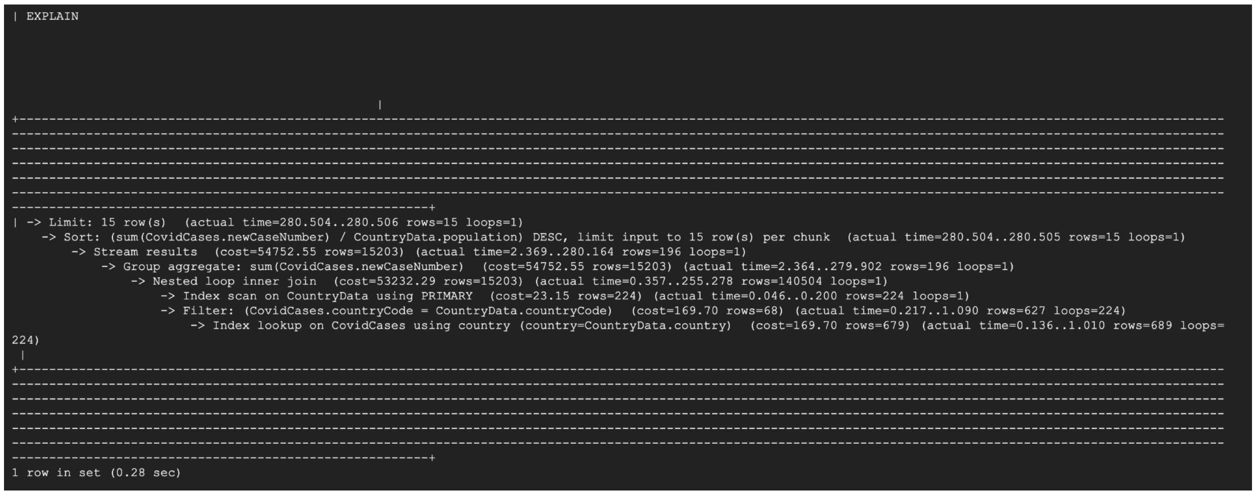
```
--------------------------------------------------------+
| EXPLAIN




                                                  |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=279.917..279.919 rows=15 loops=1)
    -> Sort: (sum(CovidCases.newCaseNumber) / CountryData.population) DESC, limit input to 15 row(s) per chunk  (actual time=279.916..279.917 rows=15 loops=1)
        -> Stream results  (cost=54752.55 rows=15203) (actual time=2.177..279.544 rows=196 loops=1)
            -> Group aggregate: sum(CovidCases.newCaseNumber)  (cost=54752.55 rows=15203) (actual time=2.172..279.250 rows=196 loops=1)
                -> Nested loop inner join  (cost=53232.29 rows=15203) (actual time=0.341..255.006 rows=140504 loops=1)
                    -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.041..0.205 rows=224 loops=1)
                    -> Filter: (CovidCases.countryCode = CountryData.countryCode)  (cost=169.70 rows=68) (actual time=0.218..1.098 rows=627 loops=224)
                        -> Index lookup on CovidCases using country (country=CountryData.country)  (cost=169.70 rows=679) (actual time=0.137..1.019 rows=689 loops=
224)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
1 row in set (0.29 sec)
```

PRIMARY and newCaseNumber index:



```
| EXPLAIN




                                                  |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=276.868..276.870 rows=15 loops=1)
    -> Sort: (sum(CovidCases.newCaseNumber) / CountryData.population) DESC, limit input to 15 row(s) per chunk  (actual time=276.867..276.868 rows=15 loops=1)
        -> Stream results  (cost=54752.55 rows=15203) (actual time=2.185..276.532 rows=196 loops=1)
            -> Group aggregate: sum(CovidCases.newCaseNumber)  (cost=54752.55 rows=15203) (actual time=2.180..276.270 rows=196 loops=1)
                -> Nested loop inner join  (cost=53232.29 rows=15203) (actual time=0.312..252.291 rows=140504 loops=1)
                    -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.034..0.192 rows=224 loops=1)
                    -> Filter: (CovidCases.countryCode = CountryData.countryCode)  (cost=169.70 rows=68) (actual time=0.216..1.086 rows=627 loops=224)
                        -> Index lookup on CovidCases using country (country=CountryData.country)  (cost=169.70 rows=679) (actual time=0.135..1.006 rows=689 loops=
224)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------+
1 row in set (0.28 sec)
```

PRIMARY, population, and newCaseNumber index:



```
| EXPLAIN




                                                  |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=280.504..280.506 rows=15 loops=1)
    -> Sort: (sum(CovidCases.newCaseNumber) / CountryData.population) DESC, limit input to 15 row(s) per chunk  (actual time=280.504..280.505 rows=15 loops=1)
        -> Stream results  (cost=54752.55 rows=15203) (actual time=2.369..280.164 rows=196 loops=1)
            -> Group aggregate: sum(CovidCases.newCaseNumber)  (cost=54752.55 rows=15203) (actual time=2.364..279.902 rows=196 loops=1)
                -> Nested loop inner join  (cost=53232.29 rows=15203) (actual time=0.357..255.278 rows=140504 loops=1)
                    -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.046..0.200 rows=224 loops=1)
                    -> Filter: (CovidCases.countryCode = CountryData.countryCode)  (cost=169.70 rows=68) (actual time=0.217..1.090 rows=627 loops=224)
                        -> Index lookup on CovidCases using country (country=CountryData.country)  (cost=169.70 rows=679) (actual time=0.136..1.010 rows=689 loops=
224)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------+
1 row in set (0.28 sec)
```

We tried two types of indexing, trying them separately and together. In short, both of them helped, but one more than the other

Our first index was on the population in the Country Data table. This affected only 224 rows however when we compared the result of this indexing to no indexing, our performance had increased by 0.07 seconds. Instead of taking 0.35 seconds, it took 0.28 seconds.

Our second index was on the newCaseNumber in the Covid Cases table - this is the largest table in our query, with approximately 158000 rows so we expected to see some results. We implemented this index after dropping our first index on population. Just by itself, it had increased our performance significantly, from 0.35 seconds to 0.28 seconds

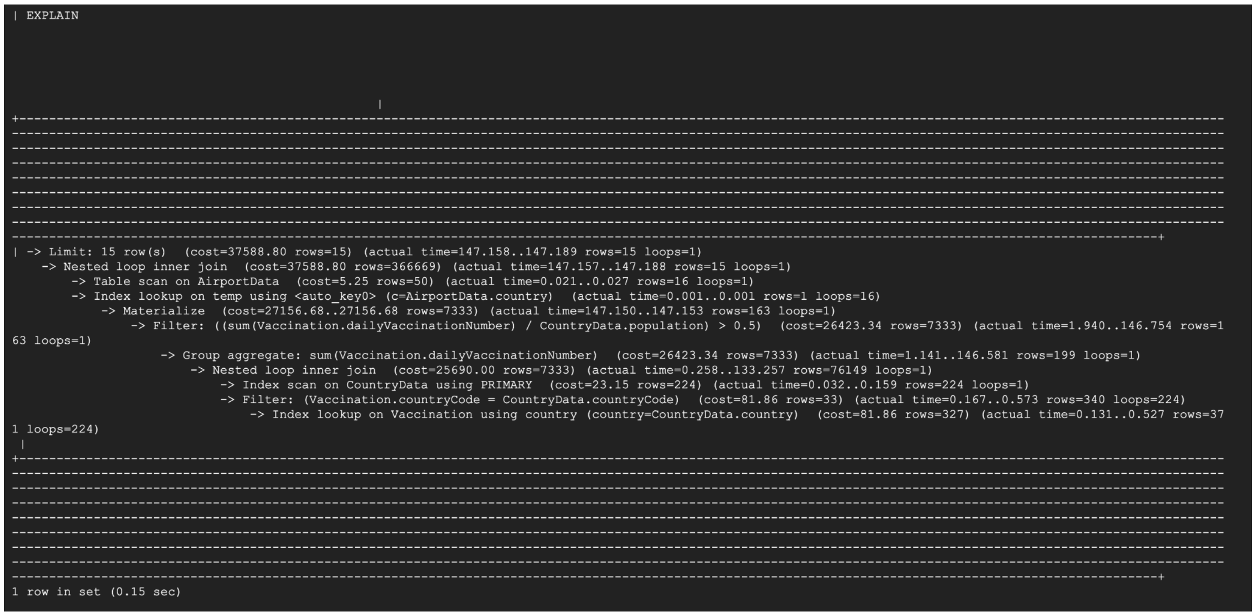
We tried both the indices together and the result was actually the same as just the second index on newCaseNumber - 0.28 seconds.

Both of them did not help increase our performance significantly. The reason it did not help was that we were not filtering or looking up rows. Furthermore, due to the query being as optimized as possible, there is no other place where an index can help. The runtime was consistently recorded when the LIMIT 15 clause was excluded in the query.

Our query groups by country which is the primary key for CountryData. This means MySQL already created the PRIMARY index for it so our original query was already fast. CovidCases.country is a foreign key but is also part of a composite primary key so it already has a composite index built on it. There is no other place to add an index so our query is as optimized as possible.

In conclusion, we will not add any new index.

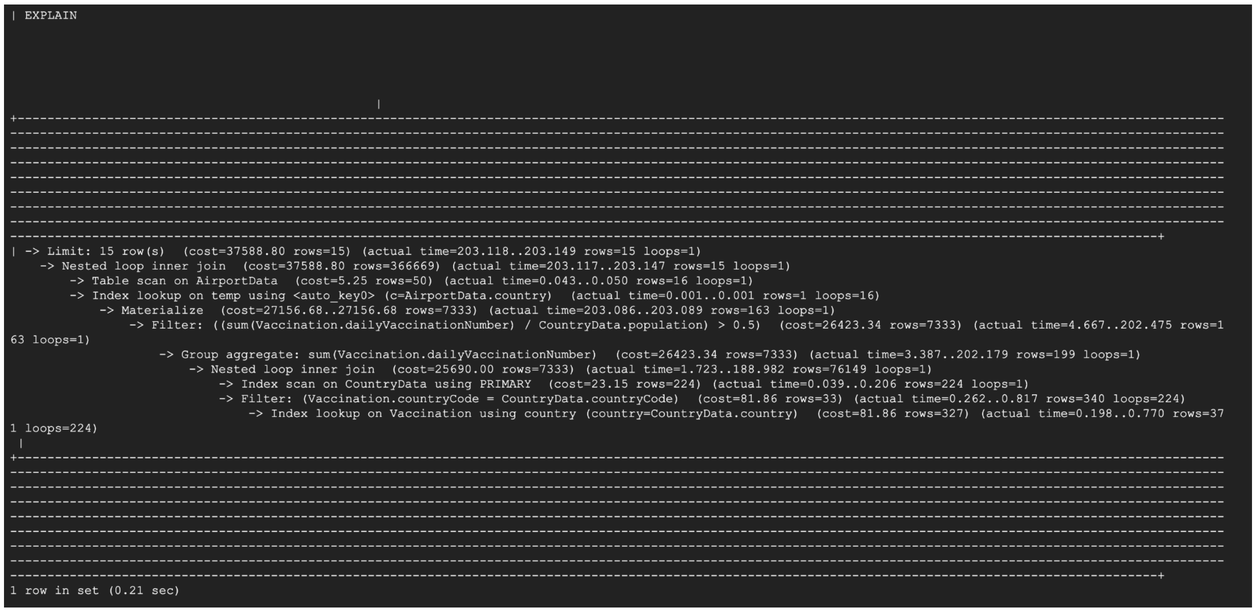
<u>Second advanced SQL command:</u>
PRIMARY index:



```
| EXPLAIN



                                                  |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=37588.80 rows=15) (actual time=147.158..147.189 rows=15 loops=1)
    -> Nested loop inner join  (cost=37588.80 rows=366669) (actual time=147.157..147.188 rows=15 loops=1)
        -> Table scan on AirportData  (cost=5.25 rows=50) (actual time=0.021..0.027 rows=16 loops=1)
        -> Index lookup on temp using <auto_key0> (c=AirportData.country)  (actual time=0.001..0.001 rows=1 loops=16)
            -> Materialize  (cost=27156.68..27156.68 rows=7333) (actual time=147.150..147.153 rows=163 loops=1)
                -> Filter: ((sum(Vaccination.dailyVaccinationNumber) / CountryData.population) > 0.5)  (cost=26423.34 rows=7333) (actual time=1.940..146.754 rows=1
63 loops=1)
                    -> Group aggregate: sum(Vaccination.dailyVaccinationNumber)  (cost=26423.34 rows=7333) (actual time=1.141..146.581 rows=199 loops=1)
                        -> Nested loop inner join  (cost=25690.00 rows=7333) (actual time=0.258..133.257 rows=76149 loops=1)
                            -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.032..0.159 rows=224 loops=1)
                            -> Filter: (Vaccination.countryCode = CountryData.countryCode)  (cost=81.86 rows=33) (actual time=0.167..0.573 rows=340 loops=224)
                                -> Index lookup on Vaccination using country (country=CountryData.country)  (cost=81.86 rows=327) (actual time=0.131..0.527 rows=37
1 loops=224)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.15 sec)
```

—--

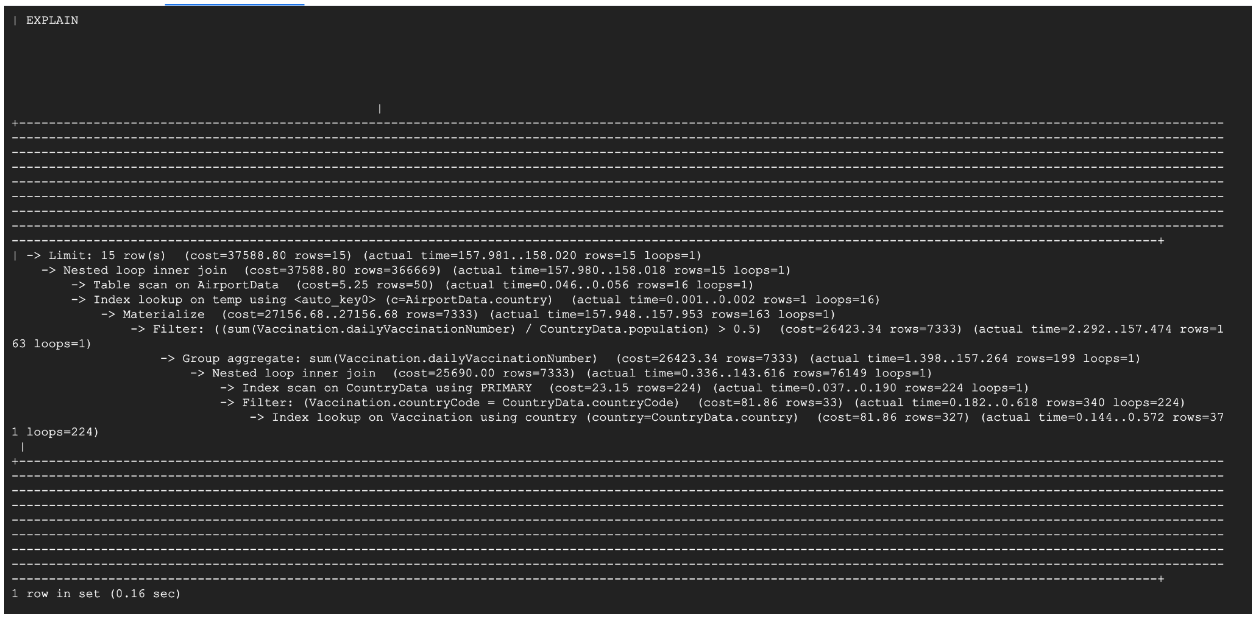
PRIMARY and population index:



```
| EXPLAIN



                                                  |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=37588.80 rows=15) (actual time=203.118..203.149 rows=15 loops=1)
    -> Nested loop inner join  (cost=37588.80 rows=366669) (actual time=203.117..203.147 rows=15 loops=1)
        -> Table scan on AirportData  (cost=5.25 rows=50) (actual time=0.043..0.050 rows=16 loops=1)
        -> Index lookup on temp using <auto_key0> (c=AirportData.country)  (actual time=0.001..0.001 rows=1 loops=16)
            -> Materialize  (cost=27156.68..27156.68 rows=7333) (actual time=203.086..203.089 rows=163 loops=1)
                -> Filter: ((sum(Vaccination.dailyVaccinationNumber) / CountryData.population) > 0.5)  (cost=26423.34 rows=7333) (actual time=4.667..202.475 rows=1
63 loops=1)
                    -> Group aggregate: sum(Vaccination.dailyVaccinationNumber)  (cost=26423.34 rows=7333) (actual time=3.387..202.179 rows=199 loops=1)
                        -> Nested loop inner join  (cost=25690.00 rows=7333) (actual time=1.723..188.982 rows=76149 loops=1)
                            -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.039..0.206 rows=224 loops=1)
                            -> Filter: (Vaccination.countryCode = CountryData.countryCode)  (cost=81.86 rows=33) (actual time=0.262..0.817 rows=340 loops=224)
                                -> Index lookup on Vaccination using country (country=CountryData.country)  (cost=81.86 rows=327) (actual time=0.198..0.770 rows=37
1 loops=224)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.21 sec)
```

—------

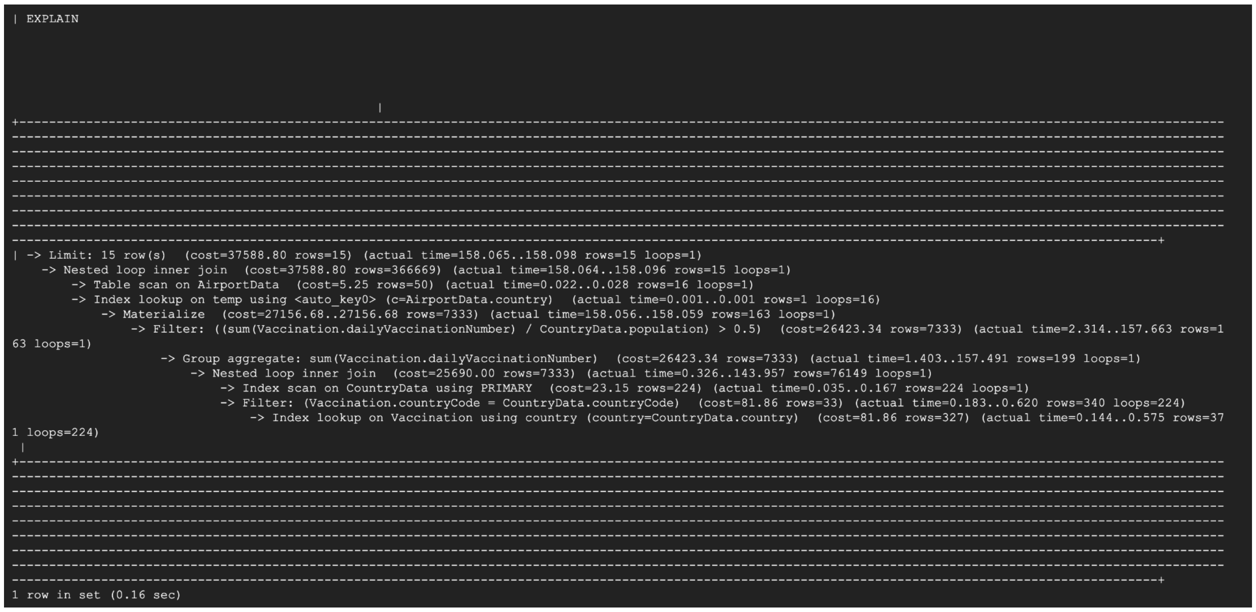
PRIMARY and dailyVaccinationNumber index:



```
| EXPLAIN



                                                  |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=37588.80 rows=15) (actual time=157.981..158.020 rows=15 loops=1)
    -> Nested loop inner join  (cost=37588.80 rows=366669) (actual time=157.980..158.018 rows=15 loops=1)
        -> Table scan on AirportData  (cost=5.25 rows=50) (actual time=0.046..0.056 rows=16 loops=1)
        -> Index lookup on temp using <auto_key0> (c=AirportData.country)  (actual time=0.001..0.002 rows=1 loops=16)
            -> Materialize  (cost=27156.68..27156.68 rows=7333) (actual time=157.948..157.953 rows=163 loops=1)
                -> Filter: ((sum(Vaccination.dailyVaccinationNumber) / CountryData.population) > 0.5)  (cost=26423.34 rows=7333) (actual time=2.292..157.474 rows=1
63 loops=1)
                    -> Group aggregate: sum(Vaccination.dailyVaccinationNumber)  (cost=26423.34 rows=7333) (actual time=1.398..157.264 rows=199 loops=1)
                        -> Nested loop inner join  (cost=25690.00 rows=7333) (actual time=0.336..143.616 rows=76149 loops=1)
                            -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.037..0.190 rows=224 loops=1)
                            -> Filter: (Vaccination.countryCode = CountryData.countryCode)  (cost=81.86 rows=33) (actual time=0.182..0.618 rows=340 loops=224)
                                -> Index lookup on Vaccination using country (country=CountryData.country)  (cost=81.86 rows=327) (actual time=0.144..0.572 rows=37
1 loops=224)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.16 sec)
```

—-----

Index on PRIMARY, population and dailyVaccinationNumber:



```
| EXPLAIN



                                                  |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=37588.80 rows=15) (actual time=158.065..158.098 rows=15 loops=1)
    -> Nested loop inner join  (cost=37588.80 rows=366669) (actual time=158.064..158.096 rows=15 loops=1)
        -> Table scan on AirportData  (cost=5.25 rows=50) (actual time=0.022..0.028 rows=16 loops=1)
        -> Index lookup on temp using <auto_key0> (c=AirportData.country)  (actual time=0.001..0.001 rows=1 loops=16)
            -> Materialize  (cost=27156.68..27156.68 rows=7333) (actual time=158.056..158.059 rows=163 loops=1)
                -> Filter: ((sum(Vaccination.dailyVaccinationNumber) / CountryData.population) > 0.5)  (cost=26423.34 rows=7333) (actual time=2.314..157.663 rows=1
63 loops=1)
                    -> Group aggregate: sum(Vaccination.dailyVaccinationNumber)  (cost=26423.34 rows=7333) (actual time=1.403..157.491 rows=199 loops=1)
                        -> Nested loop inner join  (cost=25690.00 rows=7333) (actual time=0.326..143.957 rows=76149 loops=1)
                            -> Index scan on CountryData using PRIMARY  (cost=23.15 rows=224) (actual time=0.035..0.167 rows=224 loops=1)
                            -> Filter: (Vaccination.countryCode = CountryData.countryCode)  (cost=81.86 rows=33) (actual time=0.183..0.620 rows=340 loops=224)
                                -> Index lookup on Vaccination using country (country=CountryData.country)  (cost=81.86 rows=327) (actual time=0.144..0.575 rows=37
1 loops=224)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.16 sec)
```

We tried two types of indexing, trying them separately and together. In short, neither of them helped, and one of them actually made it worse.

Our first index was on the population in the Country Data table. This affected only 224 rows so when we compared the result of this indexing to no indexing, our performance had actually decreased! Instead of taking 0.15 seconds, it took 0.21 seconds.

Our second index was on the dailyVaccinationNumber in the Vaccination table - this was the second-largest table in our query, which was approximately 74000 rows. We implemented this index after dropping our first index on population. As compared to just the PRIMARY index, there was a minute difference but this index actually decreased in performance. The PRIMARY index took 0.15 seconds and this index took 0.16 seconds.

We tried both the indices together and the result was the same as the second index - 0.16 seconds.

Our current theory for why none of them helped reduce the time was that our tables are too small. Indexing small tables is not necessarily optimal and that is what we saw here. Additionally, our current query seems to be as optimized as possible.

Our query groups by country which is the primary key for CountryData. This means MySQL already created the PRIMARY index for it so our original query was already very fast.

In conclusion, we will only index on the PRIMARY key, which is what we had originally, as it is faster than our other indices.