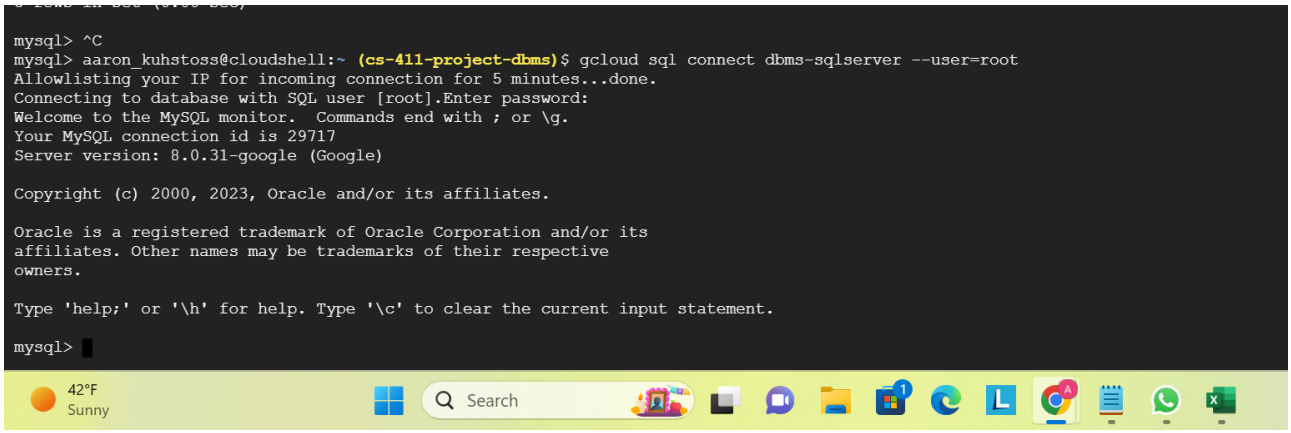
Part 1

We successfully set up our SQL instance on GCP:



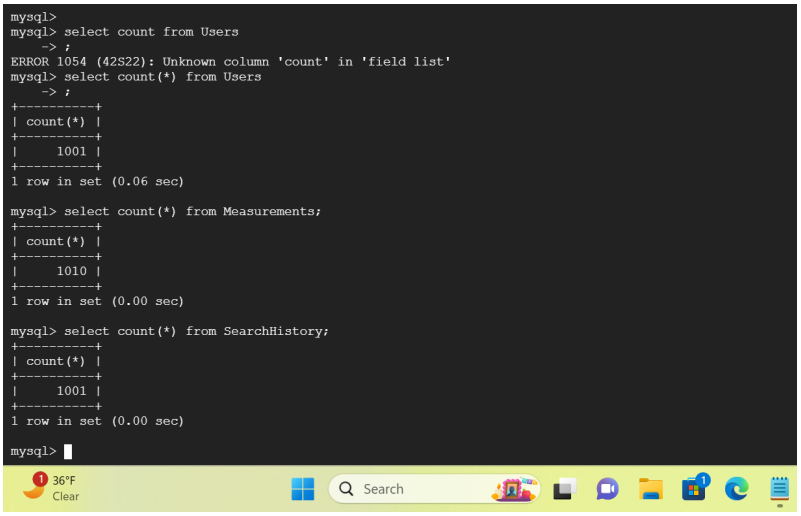
DDL Commands Used to Make Tables Below (also see repo):

```
create table Users (
        UserID int primary key,
        Username varchar(20) not null,
        UserPassword varchar(20) not null,
        UserRole varchar(10) default 'Viewer'
);

create table Location (
    SiteNum int primary key,
    CountyCode int,
    StateCode int,
    City varchar(25),
    County varchar(25),
    State varchar(25)
);

create table SearchHistory (
        SearchID int primary key,
        UserID int,
        SearchQuery varchar(64),
        foreign key(UserID) references Users(UserID)
                on delete cascade
                on update cascade
);

create table Review (
```

```
        ReviewID int primary key,
        UserID int,
        ReviewText varchar(255),
        ReviewTimestamp Timestamp,
        SiteNum int,
        ReviewDate Date,
        foreign key (UserID) references Users(UserID)
                on delete cascade
                on update cascade,
        foreign key (SiteNum) references Location(SiteNum)
                on delete restrict
                on update cascade
);


create table Measurements (
        MeasurementID int primary key,
        SiteNum int,
        MeasureDate date,
        No2Mean decimal,
        O3Mean decimal,
        So2Mean decimal,
        CoMean decimal,
        foreign key (SiteNum) references Location(SiteNum)
                on delete restrict
                on update cascade
);
```

We created our tables and populated three of them with entries; Measurements were from our data, while User and SearchHistory were generated with dummy Users and Searches.

Below is a screenshot showing that the counts for three of our tables are 1000+:

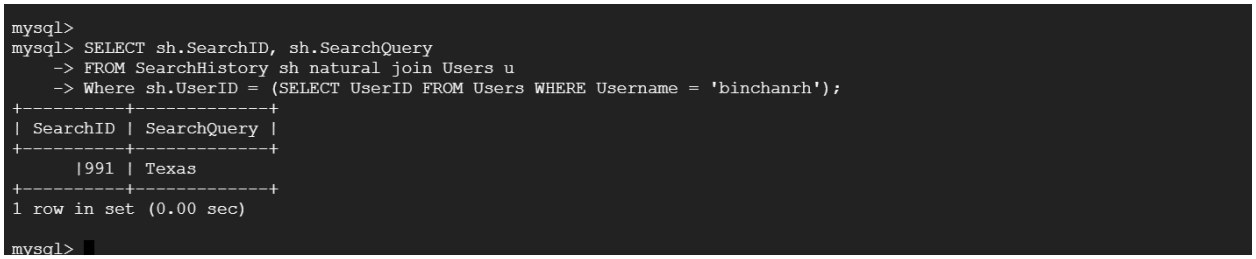


Advanced Query 1:

We want to be able to return the search query history for a given username. In this example we pick a username in our database and look for their search history using the following query.

SELECT sh.SearchID, sh.SearchQuery
FROM SearchHistory sh natural join Users u
WHERE sh.UserID = (SELECT UserID FROM Users WHERE Username = ‘binchanrh’);

Result for this query:



Notice that we only have one result here, as in our current database this user has only made one search, for “Texas”.

Advanced Query 2:

We will also want our application to be able to return the historical high and low points for a given pollutant in a given location. This will be accomplished using a query of the form below.

((select City, County, No2Mean
from Measurements natural join Location
where City = “Phoenix” and No2Mean is not null

order by No2Mean desc
limit 10)
union all
(select City, County, No2Mean
from Measurements natural join Location
where City = “Phoenix” and No2Mean is not null
order by No2Mean
limit 5));

Result for this query:

```
mysql> ((select City, County, No2Mean from Measurements natural join Location where City = 'P
on all (select City, County, No2Mean from Measurements natural join Location where City = "Ph
+---------+----------+---------+
| City    | County   | No2Mean |
+---------+----------+---------+
| Phoenix | Maricopa |      73 |
| Phoenix | Maricopa |      73 |
| Phoenix | Maricopa |      73 |
| Phoenix | Maricopa |      73 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |      67 |
| Phoenix | Maricopa |       1 |
| Phoenix | Maricopa |       1 |
| Phoenix | Maricopa |       1 |
| Phoenix | Maricopa |       1 |
| Phoenix | Maricopa |       1 |
+---------+----------+---------+
15 rows in set (0.01 sec)
```

Here the query returns the top 10 and bottom 5 measurements at the given location (in this case, Phoenix).

Part 2

Indexing:

Using EXPLAIN ANALYZE on Query 1 yields the following:

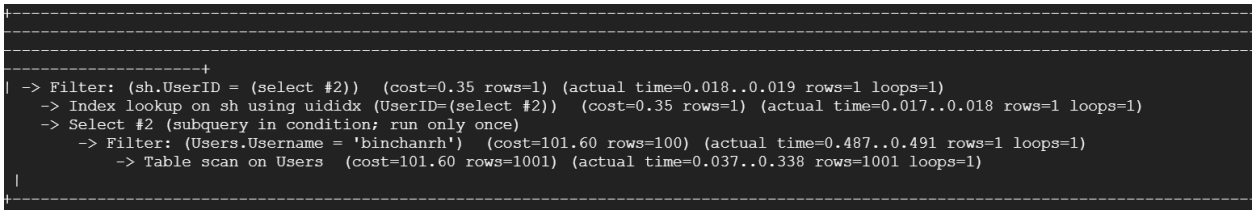
```
--------------------------------------------------------------------------------------
-------------------------------------------------------------------+
| -> Filter: (sh.UserID = (select #2))  (cost=0.35 rows=1) (actual time=0.096..0.098 ro
ws=1 loops=1)
    -> Index lookup on sh using UserID (UserID=(select #2))  (cost=0.35 rows=1) (actual
 time=0.066..0.069 rows=1 loops=1)
    -> Select #2 (subquery in condition; run only once)
        -> Filter: (Users.Username = 'binchanrh')  (cost=101.60 rows=100) (actual time=
0.625..0.629 rows=1 loops=1)
            -> Table scan on Users  (cost=101.60 rows=1001) (actual time=0.076..0.446 r
ows=1001 loops=1)
 |
+-------------------------------------------------------------------------------------
```

Here we can see our userID (primary key) index is used to accomplish the first stage fairly cheaply. It is the tablescan stage that produces the largest cost (101.60 for table scan compared to 0.35 for first).

If we try indexing on Users.Username (usernameIdx) we can dramatically reduce the cost of searching for our Username of interest, getting it down to 0.72 for the Users lookup. Creating an index on SearchHistory.UserID (uididx) does not have any effect on cost, and indexing on Users.Username and SearchHistory.UserID does not produce a cost result any different than using only the Username index. It's clear that usernameIdx is extremely effective at reducing the cost of the overall query, reducing it by nearly 100. However, it is less obvious whether it's worth it to keep the uididx index on UserID, because the nature of our current data is such that this query only has one row, and so there may be an effect with a larger search history. For now, we will keep the usernameIdx as our final indexing design for this query, and apply further analysis later on to assess the continued validity of this design as our database grows.

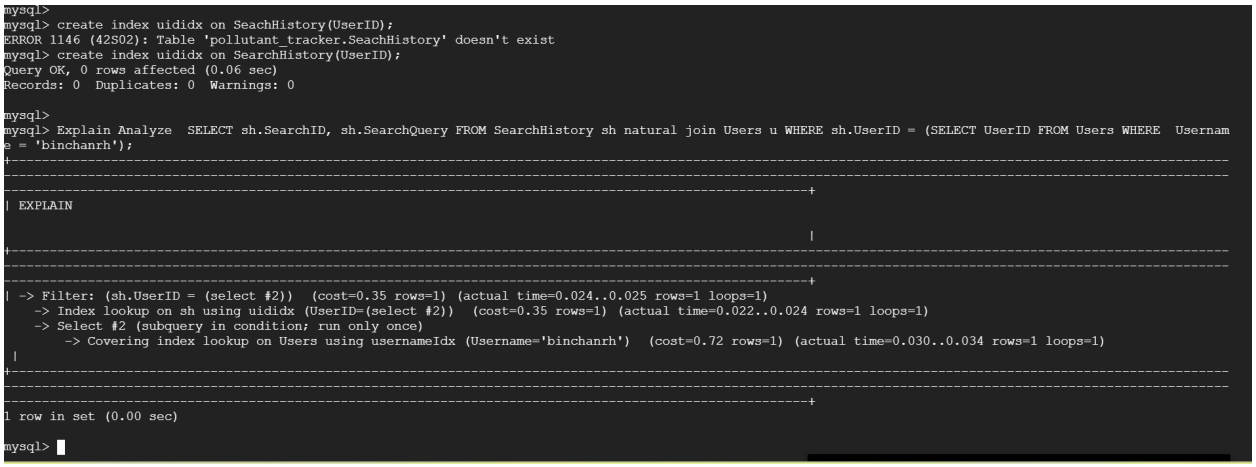
```
mysql>
mysql>
mysql>
mysql> create index usernameIdx on Users(Username);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Explain Analyze  SELECT sh.SearchID, sh.SearchQuery FROM SearchHistory sh natural join Users u WHERE sh.UserID = (SELECT UserID FROM Users WHERE  Usernam
e = 'binchanrh');
+---------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                            |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------+
| -> Filter: (sh.UserID = (select #2))  (cost=0.35 rows=1) (actual time=0.014..0.014 rows=1 loops=1)
    -> Index lookup on sh using UserID (UserID=(select #2))  (cost=0.35 rows=1) (actual time=0.013..0.014 rows=1 loops=1)
    -> Select #2 (subquery in condition; run only once)
        -> Covering index lookup on Users using usernameIdx (Username='binchanrh')  (cost=0.72 rows=1) (actual time=0.019..0.020 rows=1 loops=1)
 |
+---------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql>
```

EXPLAIN ANALYZE for Index on Users(Username)



```
| -> Filter: (sh.UserID = (select #2))  (cost=0.35 rows=1) (actual time=0.018..0.019 rows=1 loops=1)
    -> Index lookup on sh using uididx (UserID=(select #2))  (cost=0.35 rows=1) (actual time=0.017..0.018 rows=1 loops=1)
    -> Select #2 (subquery in condition; run only once)
        -> Filter: (Users.Username = 'binchanrh')  (cost=101.60 rows=100) (actual time=0.487..0.491 rows=1 loops=1)
            -> Table scan on Users  (cost=101.60 rows=1001) (actual time=0.037..0.338 rows=1001 loops=1)
 |
```

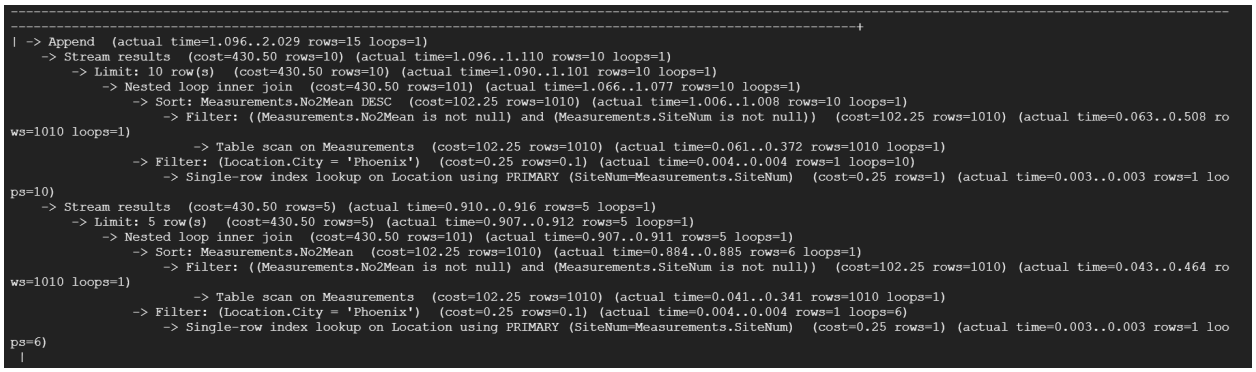
EXPLAIN ANALYZE for Index on SearchHistory(UserID)



```
mysql>
mysql> create index uididx on SeachHistory(UserID);
ERROR 1146 (42S02): Table 'pollutant_tracker.SeachHistory' doesn't exist
mysql> create index uididx on SearchHistory(UserID);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> Explain Analyze  SELECT sh.SearchID, sh.SearchQuery FROM SearchHistory sh natural join Users u WHERE sh.UserID = (SELECT UserID FROM Users WHERE  Usernam
e = 'binchanrh');
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                                                                                |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------+
| -> Filter: (sh.UserID = (select #2))  (cost=0.35 rows=1) (actual time=0.024..0.025 rows=1 loops=1)
    -> Index lookup on sh using uididx (UserID=(select #2))  (cost=0.35 rows=1) (actual time=0.022..0.024 rows=1 loops=1)
    -> Select #2 (subquery in condition; run only once)
        -> Covering index lookup on Users using usernameIdx (Username='binchanrh')  (cost=0.72 rows=1) (actual time=0.030..0.034 rows=1 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

EXPLAIN ANALYZE with Index on SearchHistory(UserID) AND Index on Users(Username)

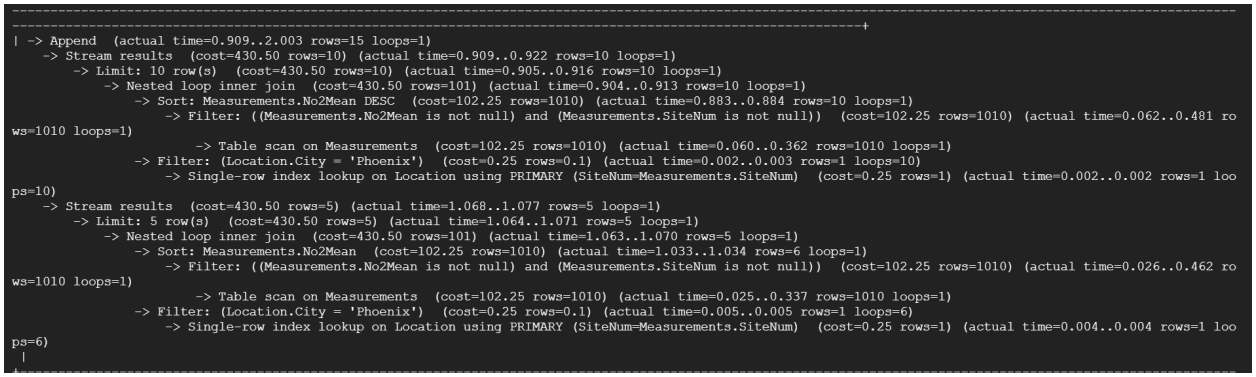
Using EXPLAIN ANALYZE on query 2 returns the following:



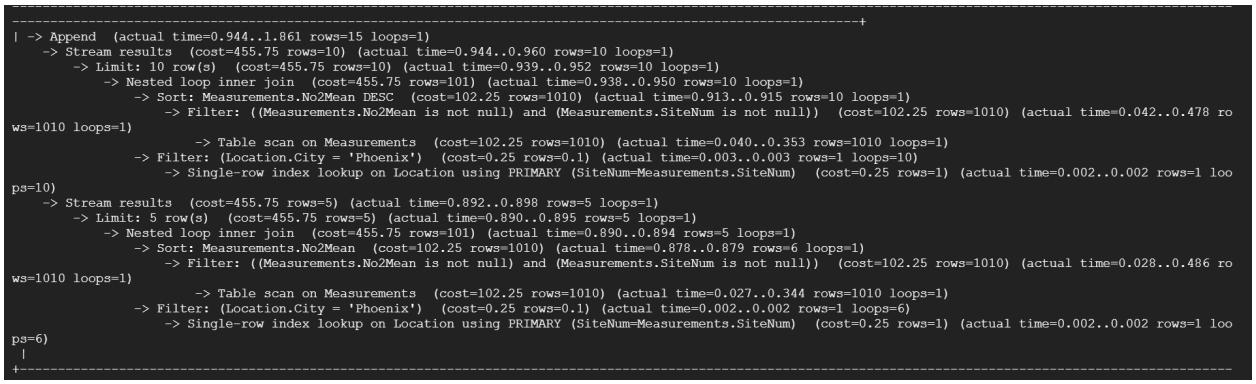
```
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------+
| -> Append  (actual time=1.096..2.029 rows=15 loops=1)
    -> Stream results  (cost=430.50 rows=10) (actual time=1.096..1.110 rows=10 loops=1)
        -> Limit: 10 row(s)  (cost=430.50 rows=10) (actual time=1.090..1.101 rows=10 loops=1)
            -> Nested loop inner join  (cost=430.50 rows=101) (actual time=1.066..1.077 rows=10 loops=1)
                -> Sort: Measurements.No2Mean DESC  (cost=102.25 rows=1010) (actual time=1.006..1.008 rows=10 loops=1)
                    -> Filter: ((Measurements.No2Mean is not null) and (Measurements.SiteNum is not null))  (cost=102.25 rows=1010) (actual time=0.063..0.508 ro
ws=1010 loops=1)
                        -> Table scan on Measurements  (cost=102.25 rows=1010) (actual time=0.061..0.372 rows=1010 loops=1)
                -> Filter: (Location.City = 'Phoenix')  (cost=0.25 rows=0.1) (actual time=0.004..0.004 rows=1 loops=10)
                    -> Single-row index lookup on Location using PRIMARY (SiteNum=Measurements.SiteNum)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loo
ps=10)
    -> Stream results  (cost=430.50 rows=5) (actual time=0.910..0.916 rows=5 loops=1)
        -> Limit: 5 row(s)  (cost=430.50 rows=5) (actual time=0.907..0.912 rows=5 loops=1)
            -> Nested loop inner join  (cost=430.50 rows=101) (actual time=0.907..0.911 rows=5 loops=1)
                -> Sort: Measurements.No2Mean  (cost=102.25 rows=1010) (actual time=0.884..0.885 rows=6 loops=1)
                    -> Filter: ((Measurements.No2Mean is not null) and (Measurements.SiteNum is not null))  (cost=102.25 rows=1010) (actual time=0.043..0.464 ro
ws=1010 loops=1)
                        -> Table scan on Measurements  (cost=102.25 rows=1010) (actual time=0.041..0.341 rows=1010 loops=1)
                -> Filter: (Location.City = 'Phoenix')  (cost=0.25 rows=0.1) (actual time=0.004..0.004 rows=1 loops=6)
                    -> Single-row index lookup on Location using PRIMARY (SiteNum=Measurements.SiteNum)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loo
ps=6)
 |
```

We can see that initially the cost of query 2 is spread over quite a few steps, including filtering, sorting, and joins. The joins are the most expensive operations in the query. Applying an index sitenum_idx to Measurements(SiteNum) once again offers no improvement in costs. Applying an index to Measurements(No2Mean) (no2mean_idx) by contrast, causes an *increase* in cost from 430.05 to 455.75. Again, we assume that this can be attributed to the smaller scale of our current database. A database containing hundreds of thousands or even millions of records may behave differently. Indexing on Location(City, County) (citycounty_idx) also shows no cost change. Based on this analysis alone, none of the attempted indexes are worth the cost of

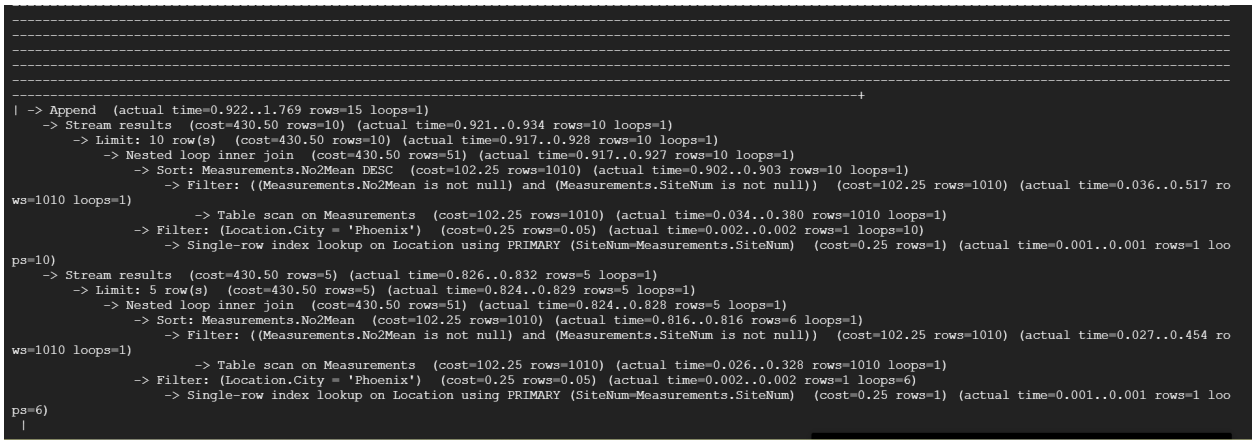
implementing. Our indexing plan will be to remain with the default indexing scheme for the query.



EXPLAIN ANALYZE for sitenum_idx on Measurements(SiteNum)



EXPLAIN ANALYZE for no2mean_idx on Measurements(No2Mean)



EXPLAIN ANALYZE for citycounty_idx on Location(City,County)