Assignment 3 Report

Manik Jain
(2022MCS2832)

Sagar Agrawal
(2022MCS2065)

Vinit Chandak
(2022EET2109)

18/04/2023

—

Introduction to Database Systems

(COL 362/COL 632)

—

Prof. Srikanta Bedathur

Queries

Queries are present in queries.sql file. We've written 88 queries which will more or less drive the entire application. Note that depending on the final implementation of the front end, there may be minute deviation from these queries. We might also add some queries if needed.

```
--Queries are with demo values, we would fetch the actual values form the frontend based
on the user input--
--We might not use all of these queries in the final project, but we have included them
for reference--
--Also, we might add new queries or slightly modify the existing ones in the future as
needed--

--Average Temperature:--
    --To populate dropdown options--
        SELECT DISTINCT state FROM average_temperature;

        SELECT DISTINCT year FROM average_temperature WHERE state = 'Maharashtra';

        with month_to_num(month, num) AS (SELECT month, case when month='January' then 1
else case when month = 'February' then 2 else case when month = 'March' then 3 else case
when month = 'April' then 4 else case when month = 'May' then 5 else case when month =
'June' then 6 else case when month = 'July' then 7 else case when month = 'August' then 8
else case when month = 'September' then 9 else case when month = 'October' then 10 else
case when month = 'November' then 11 else case when month = 'December' then 12 end end end
end end end end end end end end end AS num FROM average_temperature ORDER BY num),
        sorted(month) AS (SELECT DISTINCT average_temperature.month, num FROM
average_temperature, month_to_num WHERE state = 'Maharashtra' and year = '2000' and
average_temperature.month = month_to_num.month ORDER BY month_to_num.num)
        SELECT month FROM sorted ORDER BY num;

    --For dropdown:--
        SELECT * FROM average_temperature where state = 'Maharashtra';
        SELECT * FROM average_temperature where state = 'Maharashtra' and year = '2000';
        SELECT * FROM average_temperature where state = 'Maharashtra' and year = '2000'
and month = 'August';

    --For dropdown and range:--
        SELECT * FROM average_temperature where state = 'Maharashtra' and year between
'2000' and '2001';
```

```
    --For the entire avg_temp dataset:--
        SELECT * FROM average_temperature ORDER BY state, year;

    --For corelation between various attributes:--
        --For average temperature and state co2--
            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;

            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
ORDER BY A.state, A. year;

            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
and A.year = '1985' ORDER BY A.state, A. year;


        --For average temperature and rainfall--
            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year
ORDER BY C.state, D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' ORDER BY C.state, D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
```

```
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' and D.subdivision = 'East Uttar Pradesh' ORDER BY C.state,
D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' and D.subdivision = 'East Uttar Pradesh' and C.year = '1985'
ORDER BY C.state, D.subdivision, C.year;

--State_CO2--
    --To populate the dropdown:--
        SELECT DISTINCT state FROM state_co2;
        SELECT DISTINCT year FROM state_co2 WHERE state = 'Maharashtra';

    --For dropdown:--
        SELECT * FROM state_co2 where state = 'Maharashtra';
        SELECT * FROM state_co2 where state = 'Maharashtra' and year = '2000';

    --For dropdown and range:--
        SELECT * FROM state_co2 where state = 'Maharashtra' and year between '1999' and
'2000';

    --For the entire state_co2 dataset:--
        SELECT * FROM state_co2 ORDER BY state, year;

    --For corelation between various attributes:--
        --For average temperature and state co2--
            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;

            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
ORDER BY A.state, A. year;

            with avg_yearly_temp(avg_temp, state, year) AS (SELECT
avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
```

```
            SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
and A.year = '1985' ORDER BY A.state, A. year;

        --For State_CO2 and rainfall--
            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision)
            SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM
rainfall_state AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year ORDER BY
A.state, A.subdivision, A.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision)
            SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM
rainfall_state AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year and
A.state = 'Uttar Pradesh' ORDER BY A.state, A.subdivision, A.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision)
            SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM
rainfall_state AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year and
A.state = 'Uttar Pradesh' and A.subdivision = 'East Uttar Pradesh' ORDER BY A.state,
A.subdivision, A.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision)
            SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM
rainfall_state AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year and
A.state = 'Uttar Pradesh' and A.subdivision = 'East Uttar Pradesh' and A.year = '1985'
ORDER BY A.state, A.subdivision, A.year;

--CO2--
    --To populate the dropdown:--
        SELECT DISTINCT year FROM co2;

    --For dropdown:--
        SELECT * FROM co2 where year = '2000';

    --For dropdown and range:--
        SELECT * FROM co2 where year between '2020' and '2021';

    --For the entire co2 dataset:--
        SELECT * FROM co2 ORDER BY year;
    --For different attributes--
        SELECT co2_per_capita, year, gdp FROM co2 ORDER BY year;
        SELECT land_use_change_co2_per_capita, year, gdp FROM co2 ORDER BY year;

--Sealevel--
    --To poulate dropdown options--
        SELECT DISTINCT state FROM sealevel;

        SELECT DISTINCT sea_shore_city FROM sealevel WHERE state = 'Maharashtra';

        SELECT DISTINCT year FROM sealevel WHERE state = 'Maharashtra' and sea_shore_city
= 'Bombay';

        with month_to_num(month, num) AS (SELECT month, case when month='January' then 1
else case when month = 'February' then 2 else case when month = 'March' then 3 else case
when month = 'April' then 4 else case when month = 'May' then 5 else case when month =
'June' then 6 else case when month = 'July' then 7 else case when month = 'August' then 8
else case when month = 'September' then 9 else case when month = 'October' then 10 else
case when month = 'November' then 11 else case when month = 'December' then 12 end end end
end end end end end end end end end AS num FROM sealevel ORDER BY num),
        sorted(month) AS (SELECT DISTINCT sealevel.month, num FROM sealevel, month_to_num
WHERE state = 'Maharashtra' and sea_shore_city = 'Bombay' and year = '2006' and
sealevel.month = month_to_num.month ORDER BY month_to_num.num)
        SELECT month FROM sorted ORDER BY num;

    --For dropdown:--
        SELECT * FROM sealevel where state = 'Maharashtra';

        SELECT * FROM sealevel where state = 'Maharashtra' and sea_shore_city = 'Bombay';

        SELECT * FROM sealevel where state = 'Maharashtra' and sea_shore_city = 'Bombay'
and year = '2006';

        SELECT * FROM sealevel where state = 'Maharashtra' and sea_shore_city = 'Bombay'
and year = '2006' and month = 'January';

    --For dropdown and range:--
        SELECT * FROM sealevel where state = 'Maharashtra' and sea_shore_city = 'Bombay'
and year between '1990' and '2006';

        SELECT * FROM sealevel where state = 'Maharashtra' and sea_shore_city = 'Bombay'
and year between '1990' and '2006' and month between 'June' and 'September';

    --For the entire sealevel dataset:--
        SELECT * FROM sealevel ORDER BY state, sea_shore_city, year, month;
```

```sql
--AQI--
    --To populate dropdown options--
        SELECT DISTINCT State FROM AQI;
        SELECT DISTINCT City FROM AQI WHERE State='Gujarat';
    --For dropdown--
        SELECT * FROM AQI WHERE State ='Gujarat';
        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad';

        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015';

        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01';

        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE)='01';

    --For dropdown and range--
        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE) BETWEEN '2015' AND '2016';

        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE) BETWEEN '01' AND '03';

        SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE) BETWEEN '1' AND '15';

    --For the entire AQI dataset--
        SELECT * FROM AQI ORDER BY State,City,Date;

    --For different attributes--
        SELECT city, state, date, pm2_5 FROM AQI ORDER BY state, city, date;
        SELECT city, state, date, pm10 FROM AQI ORDER BY state, city, date;
        SELECT city, state, date, aqi FROM AQI ORDER BY state, city, date;

--Rainfall--
    --To populate the dropdown options--
        SELECT DISTINCT Subdivision FROM Rainfall;
        SELECT DISTINCT Year FROM Rainfall WHERE Subdivision='Sikkim';
    --For dropdown--
        SELECT * FROM Rainfall WHERE Subdivision='Sikkim';
        SELECT * FROM Rainfall WHERE Subdivision='Sikkim' AND Year='1947';
    --For dropdown and range--
        SELECT * FROM Rainfall WHERE Subdivision='Sikkim' AND Year BETWEEN '1947' AND '1970';
    --For the entire rainfall dataset--
        SELECT * FROM Rainfall ORDER BY Subdivision,Year;
    --For different attributes--
        SELECT subdivision, year, annual FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, january_february FROM Rainfall ORDER BY subdivision,
year;
        SELECT subdivision, year, march_may FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, june_september FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, october_december FROM Rainfall ORDER BY subdivision,
year;
        SELECT subdivision, year, January FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, February FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, March FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, April FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, May FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, June FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, July FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, August FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, September FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, October FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, November FROM Rainfall ORDER BY subdivision, year;
        SELECT subdivision, year, December FROM Rainfall ORDER BY subdivision, year;
        --For each of these queries we can have ranges? and selections?--

    --For correlation between various attributes--
        --For average temperature and rainfall--
            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year
ORDER BY C.state, D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' ORDER BY C.state, D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
```

```
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' and D.subdivision = 'East Uttar Pradesh' ORDER BY C.state,
D.subdivision, C.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature),
state, year FROM average_temperature GROUP BY state, year)
            SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM
avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and
C.state = 'Uttar Pradesh' and D.subdivision = 'East Uttar Pradesh' and C.year = '1985'
ORDER BY C.state, D.subdivision, C.year;

        --For sealevel and rainfall--
            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year
FROM rainfall_state GROUP BY state, year),
            avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year,
avg(monthly_msl) FROM sealevel GROUP BY state, year)
            SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A,
avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state,
A.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year
FROM rainfall_state GROUP BY state, year),
            avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year,
avg(monthly_msl) FROM sealevel GROUP BY state, year)
            SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A,
avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Andhra
Pradesh' ORDER BY A.state, A.year;

            with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
            rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year
FROM rainfall_state GROUP BY state, year),
            avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year,
avg(monthly_msl) FROM sealevel GROUP BY state, year)
            SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A,
avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Andhra
Pradesh' and A.year = '1985' ORDER BY A.state, A.year;
```

Database Details

Size: 3 MB
Size of the database dump(db_dump_project): 3.2 MB
Number of tuples in each relation:

Relation Name	No of Tuples
AQI	29532
Average Temperature	26113
Cities	40
CO2	173
Rainfall	2557
SeaLevel	7501
State_CO2	400
State_Subdivision	41
States	37

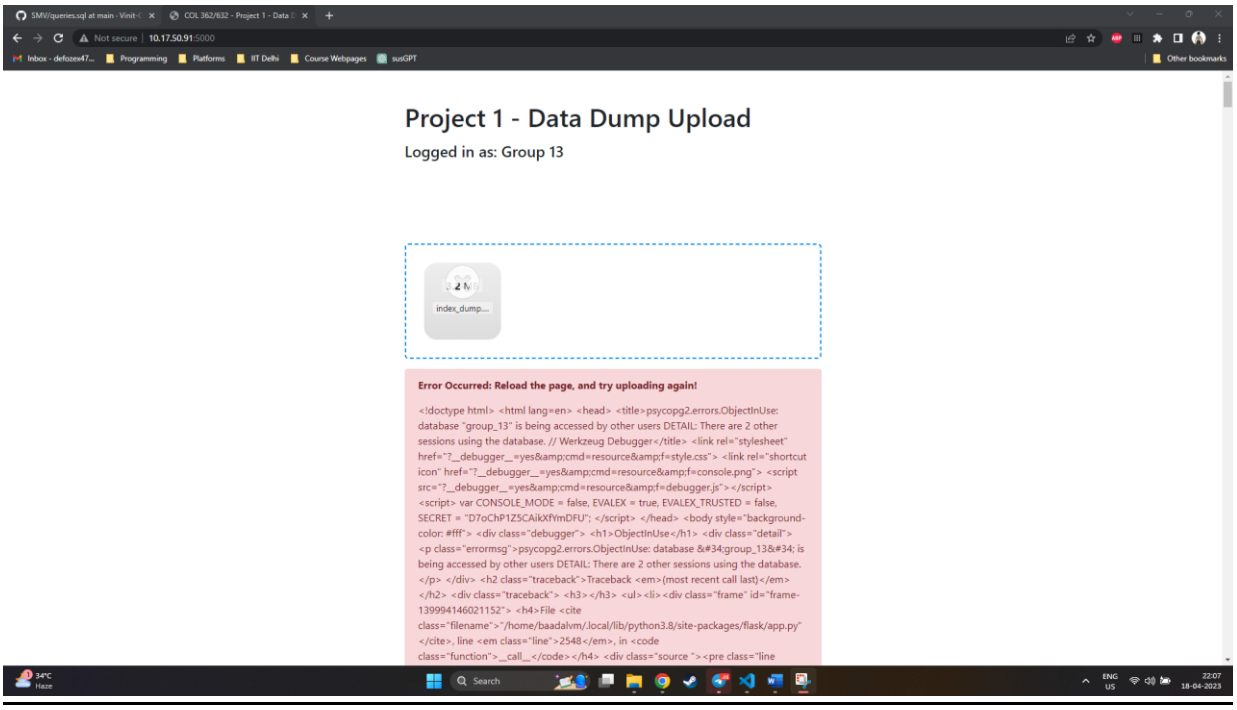
Performance Metrics

Out of the 88 queries, we're showing the performance metrics for some selected queries:

Query No	Execution Time (ms)	
	Before Indexing	After Indexing
1	18.395	0.359
2	2.469	2.2
3	39.534	28.378
4	77.191	63.487
5	32.429	32.729
6	28.251	34.270
7	18.030	2.614
8	15.258	14.723
9	3.359	1.645
10	43.931	0.896

11	9.769	7.233
12	22.696	14.540
13	21.733	17.646
14	143.347	153.990

Initially, we ran all the queries and noted the execution times on the database server instance provided to us. The after creating indexes when we tried to upload the database dump with indexes, we were repeatedly getting the following error.
We understand that this error is due to having a second ongoing connection but we verified that out other connections were terminated and still we were getting the same error.



Project 1 - Data Dump Upload

Logged in as: Group 13

Error Occurred: Reload the page, and try uploading again!

```
<!doctype html> <html lang=en> <head> <title>psycopg2.errors.ObjectInUse:
database "group_13" is being accessed by other users DETAIL: There are 2 other
sessions using the database. // Werkzeug Debugger</title> <link rel="stylesheet"
href="?__debugger__=yes&amp;cmd=resource&amp;f=style.css"> <link rel="shortcut
icon" href="?__debugger__=yes&amp;cmd=resource&amp;f=console.png"> <script
src="?__debugger__=yes&amp;cmd=resource&amp;f=debugger.js"></script>
<script> var CONSOLE_MODE = false, EVALEX = true, EVALEX_TRUSTED = false,
SECRET = "D7oChP1Z5CAikXfYmDFU"; </script> </head> <body style="background-
color: #fff"> <div class="debugger"> <h1>ObjectInUse</h1> <div class="detail">
<p class="errormsg">psycopg2.errors.ObjectInUse: database &#34;group_13&#34; is
being accessed by other users DETAIL: There are 2 other sessions using the database.
</p> </div> <h2 class="traceback">Traceback <em>(most recent call last)</em>
</h2> <div class="traceback"> <h3></h3> <ul><li><div class="frame" id="frame-
139994146021152"> <h4>File <cite
class="filename">"/home/baadalvm/.local/lib/python3.8/site-packages/flask/app.py"
</cite>, line <em class="line">2548</em>, in <code
class="function">__call__</code></h4> <div class="source "><pre class="line
```

Therefore, the metrics after indexing are based on the local database and not the server instance provided to us.

Indexes created:

idx_state_subdivision_subdivision_state ON state_subdivision(subdivision, state);

avg_temp_state_idx on average_temperature(state);

avg_temp_year_idx on average_temperature(year);
avg_temp_state_year_idx on average_temperature(state, year);

rainfall_subdiv_idx on rainfall(subdivision);
rainfall_year_idx on rainfall(year);
rainfall_state_idx on rainfall(state);
rainfall_state_year_idx on rainfall(state, year);
idx_rainfall_subdivision_annual_year ON rainfall(subdivision, annual, year);

state_co2_state_idx ON state_co2(state);
state_co2_year_idx ON state_co2(year);
state_co2_state_year_idx ON state_co2(state, year);

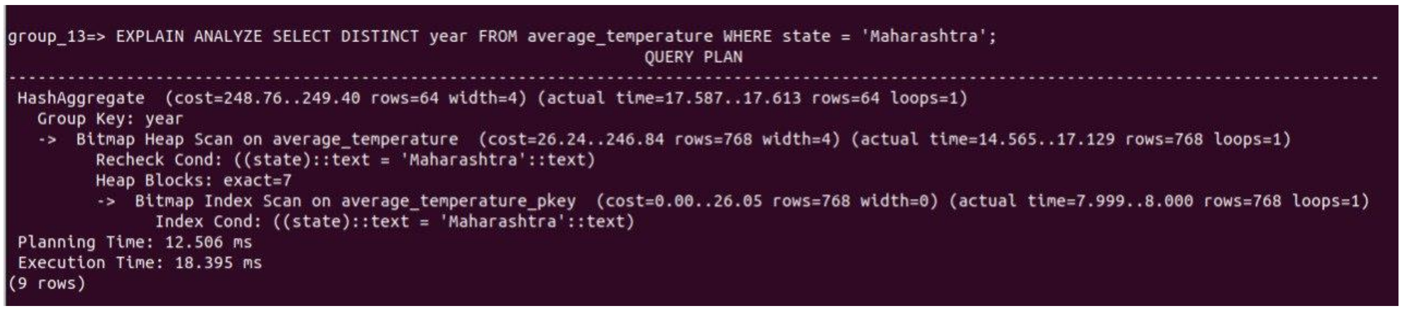
sealevel_state_idx ON sealevel(state);
sealevel_year_idx ON sealevel(year);
sealevel_city_idx ON sealevel(sea_shore_city);
idx_sealevel_state_year_monthly_msl ON sealevel(state, year, monthly_msl);

aqi_state_idx ON aqi(state);
aqi_city_idx ON aqi(city);
aqi_year_idx ON aqi(year);
aqi_date_idx ON aqi(date);
co2_year_idx ON co2(year);

Apart from this the implicit indexes created on the primary key are still present.

Query 1:

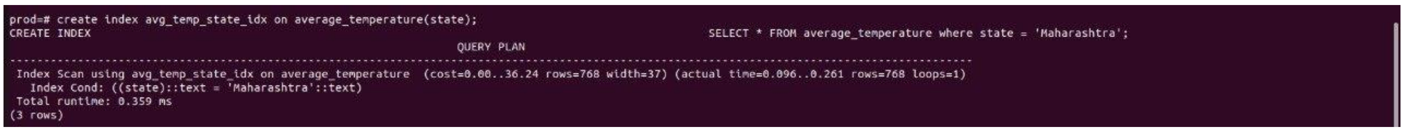
```
SELECT * FROM average_temperature where state = 'Maharashtra';
```



```
group_13=> EXPLAIN ANALYZE SELECT DISTINCT year FROM average_temperature WHERE state = 'Maharashtra';
                                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 HashAggregate  (cost=248.76..249.40 rows=64 width=4) (actual time=17.587..17.613 rows=64 loops=1)
   Group Key: year
   ->  Bitmap Heap Scan on average_temperature  (cost=26.24..246.84 rows=768 width=4) (actual time=14.565..17.129 rows=768 loops=1)
         Recheck Cond: ((state)::text = 'Maharashtra'::text)
         Heap Blocks: exact=7
         ->  Bitmap Index Scan on average_temperature_pkey  (cost=0.00..26.05 rows=768 width=0) (actual time=7.999..8.000 rows=768 loops=1)
               Index Cond: ((state)::text = 'Maharashtra'::text)
 Planning Time: 12.506 ms
 Execution Time: 18.395 ms
(9 rows)
```

Indexing:

Created index avg_temp_state_idx on state attribute of average_temperature relation. The query uses the index and the performance is vastly improved.

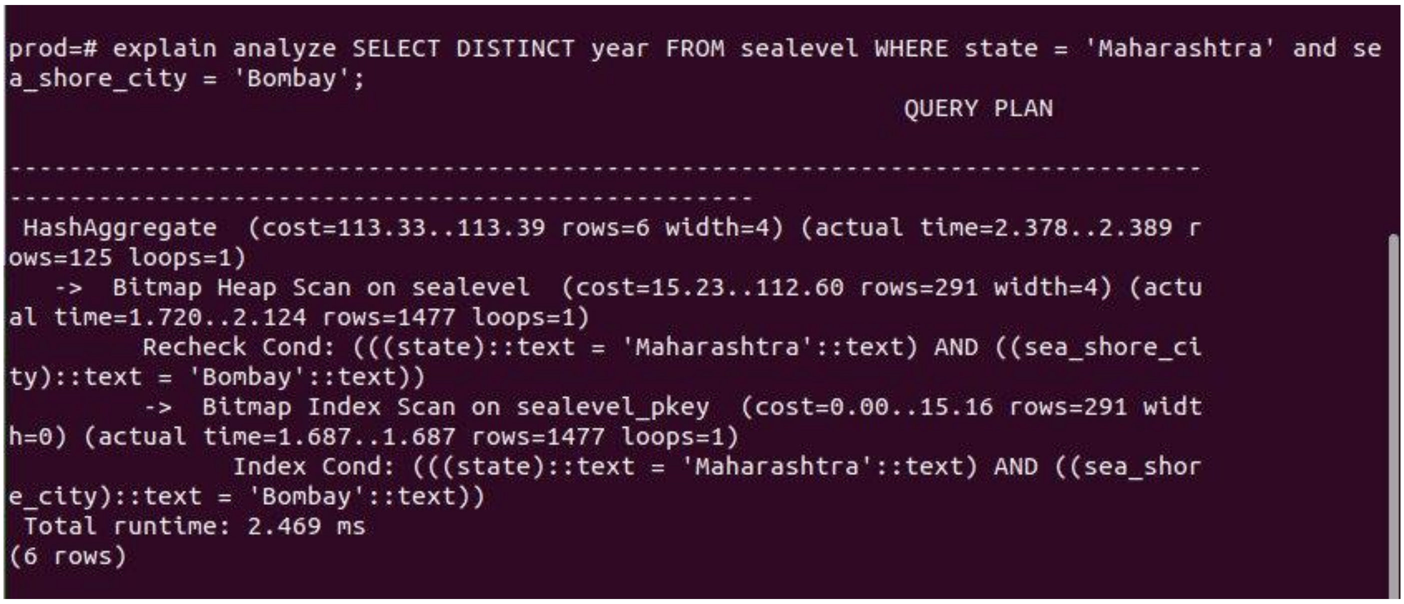


```
prod=# create index avg_temp_state_idx on average_temperature(state);
CREATE INDEX
                                                                              SELECT * FROM average_temperature where state = 'Maharashtra';
                                                      QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using avg_temp_state_idx on average_temperature  (cost=0.00..36.24 rows=768 width=37) (actual time=0.096..0.261 rows=768 loops=1)
   Index Cond: ((state)::text = 'Maharashtra'::text)
 Total runtime: 0.359 ms
(3 rows)
```

Query 2:

```
SELECT DISTINCT year FROM sealevel WHERE state = 'Maharashtra' and sea_shore_city =
'Bombay';
```

Uses the index on primary key of the sealevel table. This index was generated implicitly.

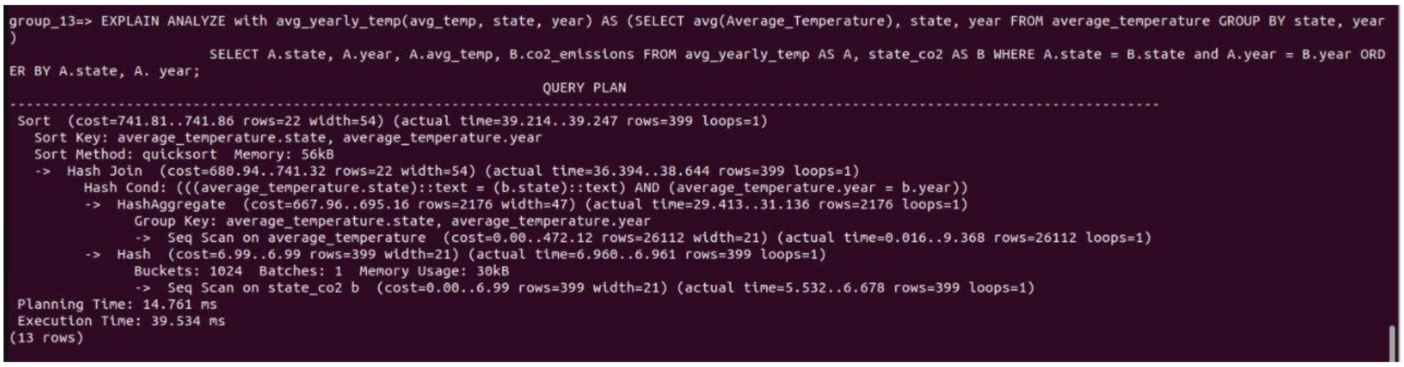


```
prod=# explain analyze SELECT DISTINCT year FROM sealevel WHERE state = 'Maharashtra' and se
a_shore_city = 'Bombay';
                                                            QUERY PLAN

--------------------------------------------------------------------------------
--------------------------------------------------
 HashAggregate  (cost=113.33..113.39 rows=6 width=4) (actual time=2.378..2.389 r
ows=125 loops=1)
   ->  Bitmap Heap Scan on sealevel  (cost=15.23..112.60 rows=291 width=4) (actu
al time=1.720..2.124 rows=1477 loops=1)
         Recheck Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shore_ci
ty)::text = 'Bombay'::text))
         ->  Bitmap Index Scan on sealevel_pkey  (cost=0.00..15.16 rows=291 widt
h=0) (actual time=1.687..1.687 rows=1477 loops=1)
               Index Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shor
e_city)::text = 'Bombay'::text))
 Total runtime: 2.469 ms
(6 rows)
```

Query 3:

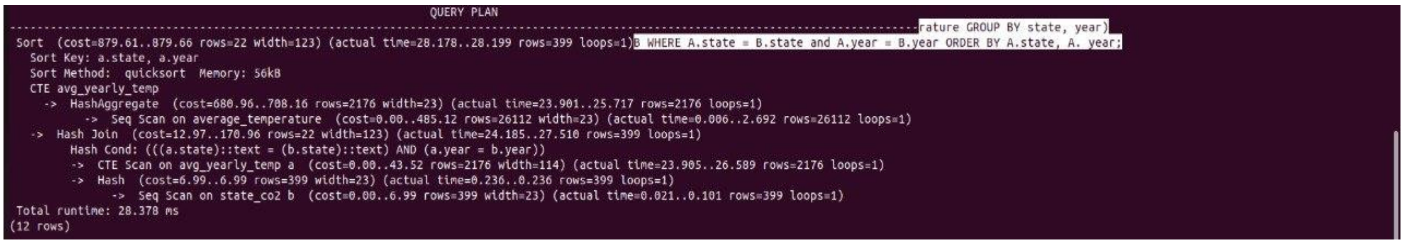
```
with avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state,
year FROM average_temperature GROUP BY state, year)
SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A, state_co2
AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;
```



```
group_13=> EXPLAIN ANALYZE with avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year
)
                          SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year ORD
ER BY A.state, A. year;
                                                                QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=741.81..741.86 rows=22 width=54) (actual time=39.214..39.247 rows=399 loops=1)
   Sort Key: average_temperature.state, average_temperature.year
   Sort Method: quicksort  Memory: 56kB
   ->  Hash Join  (cost=680.94..741.32 rows=22 width=54) (actual time=36.394..38.644 rows=399 loops=1)
         Hash Cond: (((average_temperature.state)::text = (b.state)::text) AND (average_temperature.year = b.year))
         ->  HashAggregate  (cost=667.96..695.16 rows=2176 width=47) (actual time=29.413..31.136 rows=2176 loops=1)
               Group Key: average_temperature.state, average_temperature.year
               ->  Seq Scan on average_temperature  (cost=0.00..472.12 rows=26112 width=21) (actual time=0.016..9.368 rows=26112 loops=1)
         ->  Hash  (cost=6.99..6.99 rows=399 width=21) (actual time=6.960..6.961 rows=399 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 30kB
               ->  Seq Scan on state_co2 b  (cost=0.00..6.99 rows=399 width=21) (actual time=5.532..6.678 rows=399 loops=1)
 Planning Time: 14.761 ms
 Execution Time: 39.534 ms
(13 rows)
```

Indexing:

We have an index on state attribute of the average_temperature relation, but still the index is not used by the query.

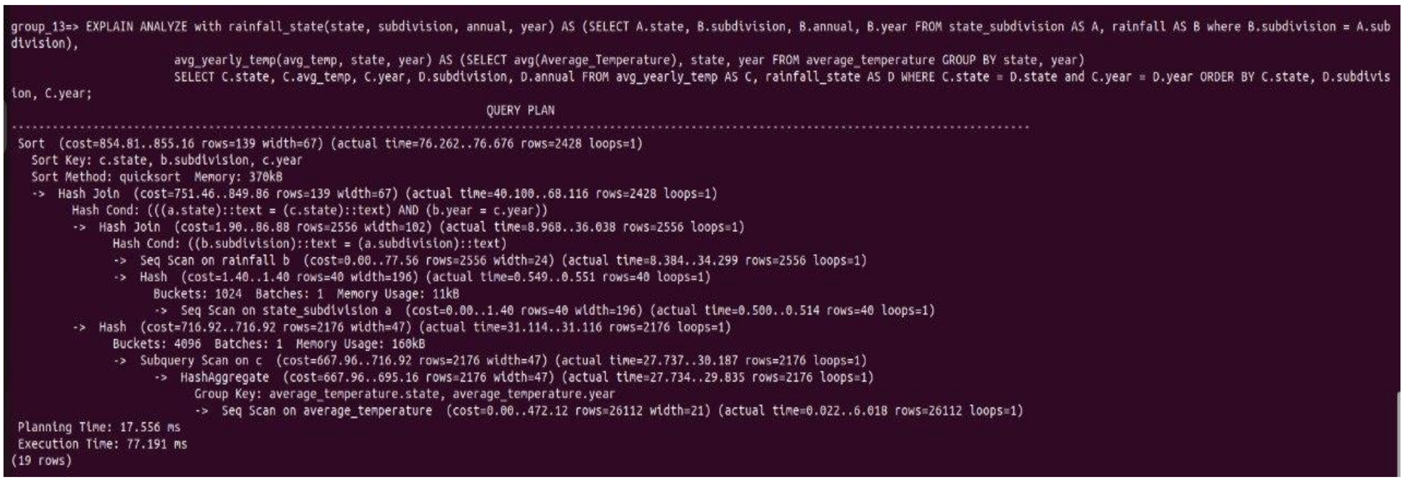


```
                                                          QUERY PLAN
----------------------------------------------------------------------------------------------------------------------rature GROUP BY state, year)
 Sort  (cost=879.61..879.66 rows=22 width=123) (actual time=28.178..28.199 rows=399 loops=1)B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;
   Sort Key: a.state, a.year
   Sort Method:  quicksort  Memory: 56kB
   CTE avg_yearly_temp
     ->  HashAggregate  (cost=680.96..708.16 rows=2176 width=23) (actual time=23.901..25.717 rows=2176 loops=1)
           ->  Seq Scan on average_temperature  (cost=0.00..485.12 rows=26112 width=23) (actual time=0.006..2.692 rows=26112 loops=1)
   ->  Hash Join  (cost=12.97..170.96 rows=22 width=123) (actual time=24.185..27.510 rows=399 loops=1)
         Hash Cond: (((a.state)::text = (b.state)::text) AND (a.year = b.year))
         ->  CTE Scan on avg_yearly_temp a  (cost=0.00..43.52 rows=2176 width=114) (actual time=23.905..26.589 rows=2176 loops=1)
         ->  Hash  (cost=6.99..6.99 rows=399 width=23) (actual time=0.236..0.236 rows=399 loops=1)
               ->  Seq Scan on state_co2 b  (cost=0.00..6.99 rows=399 width=23) (actual time=0.021..0.101 rows=399 loops=1)
 Total runtime: 28.378 ms
(12 rows)
```

Query 4:

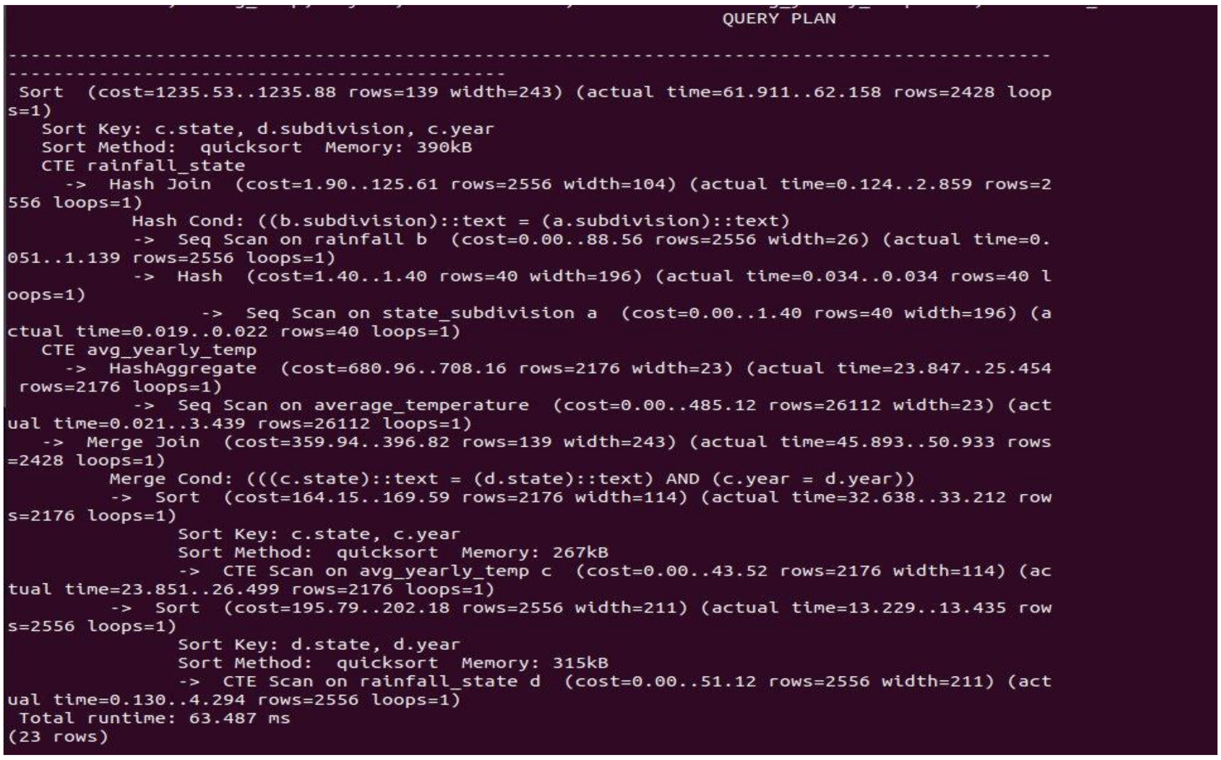
```
with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state,
B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where
B.subdivision = A.subdivision),
avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state,
year FROM average_temperature GROUP BY state, year)

SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM avg_yearly_temp AS
C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year ORDER BY C.state,
D.subdivision, C.year;
```



```
group_13=> EXPLAIN ANALYZE with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision = A.sub
division),
                        avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
                        SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year ORDER BY C.state, D.subdivis
ion, C.year;
                                                                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=854.81..855.16 rows=139 width=67) (actual time=76.262..76.676 rows=2428 loops=1)
   Sort Key: c.state, b.subdivision, c.year
   Sort Method: quicksort  Memory: 370kB
   ->  Hash Join  (cost=751.46..849.86 rows=139 width=67) (actual time=40.100..68.116 rows=2428 loops=1)
         Hash Cond: (((a.state)::text = (c.state)::text) AND (b.year = c.year))
         ->  Hash Join  (cost=1.90..86.88 rows=2556 width=102) (actual time=8.968..36.038 rows=2556 loops=1)
               Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
               ->  Seq Scan on rainfall b  (cost=0.00..77.56 rows=2556 width=24) (actual time=8.384..34.299 rows=2556 loops=1)
               ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.549..0.551 rows=40 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 11kB
                     ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.500..0.514 rows=40 loops=1)
         ->  Hash  (cost=716.92..716.92 rows=2176 width=47) (actual time=31.114..31.116 rows=2176 loops=1)
               Buckets: 4096  Batches: 1  Memory Usage: 160kB
               ->  Subquery Scan on c  (cost=667.96..716.92 rows=2176 width=47) (actual time=27.737..30.187 rows=2176 loops=1)
                     ->  HashAggregate  (cost=667.96..695.16 rows=2176 width=47) (actual time=27.734..29.835 rows=2176 loops=1)
                           Group Key: average_temperature.state, average_temperature.year
                           ->  Seq Scan on average_temperature  (cost=0.00..472.12 rows=26112 width=21) (actual time=0.022..6.018 rows=26112 loops=1)
 Planning Time: 17.556 ms
 Execution Time: 77.191 ms
(19 rows)
```

Indexing:



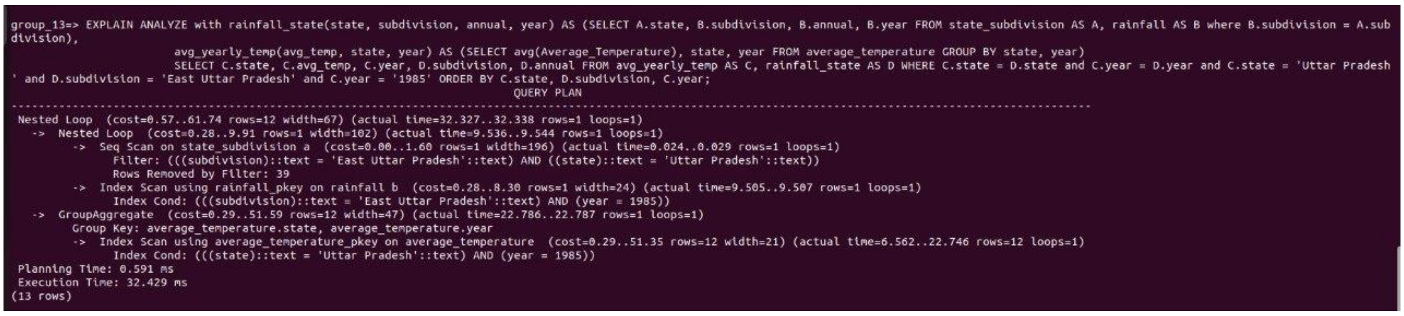
```
                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------
---------------------------------------------
 Sort  (cost=1235.53..1235.88 rows=139 width=243) (actual time=61.911..62.158 rows=2428 loop
s=1)
   Sort Key: c.state, d.subdivision, c.year
   Sort Method:  quicksort  Memory: 390kB
   CTE rainfall_state
     ->  Hash Join  (cost=1.90..125.61 rows=2556 width=104) (actual time=0.124..2.859 rows=2
556 loops=1)
           Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
           ->  Seq Scan on rainfall b  (cost=0.00..88.56 rows=2556 width=26) (actual time=0.
051..1.139 rows=2556 loops=1)
           ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.034..0.034 rows=40 l
oops=1)
                 ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (a
ctual time=0.019..0.022 rows=40 loops=1)
   CTE avg_yearly_temp
     ->  HashAggregate  (cost=680.96..708.16 rows=2176 width=23) (actual time=23.847..25.454
 rows=2176 loops=1)
           ->  Seq Scan on average_temperature  (cost=0.00..485.12 rows=26112 width=23) (act
ual time=0.021..3.439 rows=26112 loops=1)
   ->  Merge Join  (cost=359.94..396.82 rows=139 width=243) (actual time=45.893..50.933 rows
=2428 loops=1)
         Merge Cond: (((c.state)::text = (d.state)::text) AND (c.year = d.year))
         ->  Sort  (cost=164.15..169.59 rows=2176 width=114) (actual time=32.638..33.212 row
s=2176 loops=1)
               Sort Key: c.state, c.year
               Sort Method:  quicksort  Memory: 267kB
               ->  CTE Scan on avg_yearly_temp c  (cost=0.00..43.52 rows=2176 width=114) (ac
tual time=23.851..26.499 rows=2176 loops=1)
         ->  Sort  (cost=195.79..202.18 rows=2556 width=211) (actual time=13.229..13.435 row
s=2556 loops=1)
               Sort Key: d.state, d.year
               Sort Method:  quicksort  Memory: 315kB
               ->  CTE Scan on rainfall_state d  (cost=0.00..51.12 rows=2556 width=211) (act
ual time=0.130..4.294 rows=2556 loops=1)
 Total runtime: 63.487 ms
(23 rows)
```

Query 5:

```sql
with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision,
B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision =
A.subdivision),
avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state, year
FROM average_temperature GROUP BY state, year)

SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM avg_yearly_temp AS C,
rainfall_state AS D WHERE C.state = D.state and C.year = D.year and C.state = 'Uttar
```

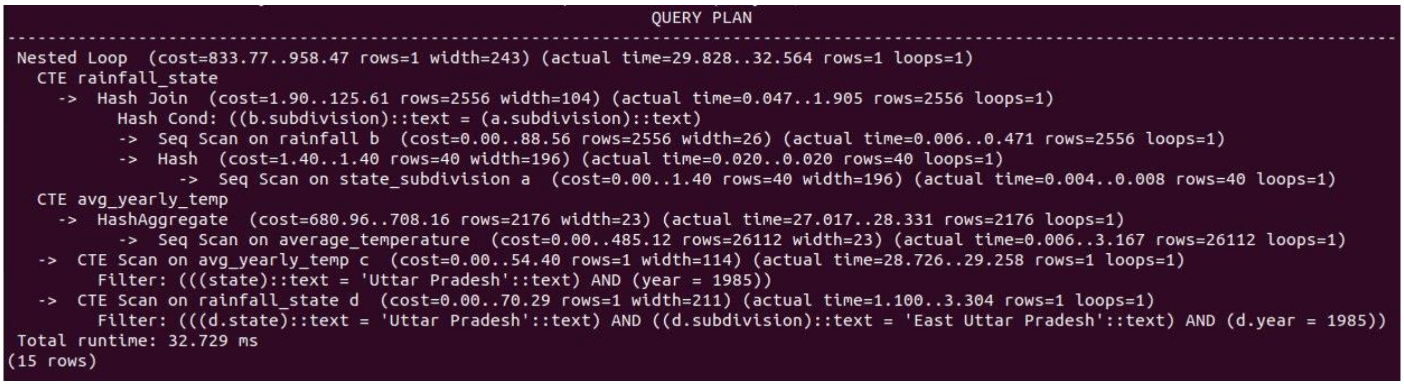
```
Pradesh' and D.subdivision = 'East Uttar Pradesh' and C.year = '1985' ORDER BY C.state,
D.subdivision, C.year;
```



```
group_13=> EXPLAIN ANALYZE with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision = A.sub
division),
                        avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
                        SELECT C.state, C.avg_temp, C.year, D.subdivision, D.annual FROM avg_yearly_temp AS C, rainfall_state AS D WHERE C.state = D.state and C.year = D.year and C.state = 'Uttar Pradesh
' and D.subdivision = 'East Uttar Pradesh' and C.year = '1985' ORDER BY C.state, D.subdivision, C.year;
                                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.57..61.74 rows=12 width=67) (actual time=32.327..32.338 rows=1 loops=1)
   ->  Nested Loop  (cost=0.28..9.91 rows=1 width=102) (actual time=9.536..9.544 rows=1 loops=1)
         ->  Seq Scan on state_subdivision a  (cost=0.00..1.60 rows=1 width=196) (actual time=0.024..0.029 rows=1 loops=1)
               Filter: (((subdivision)::text = 'East Uttar Pradesh'::text) AND ((state)::text = 'Uttar Pradesh'::text))
               Rows Removed by Filter: 39
         ->  Index Scan using rainfall_pkey on rainfall b  (cost=0.28..8.30 rows=1 width=24) (actual time=9.505..9.507 rows=1 loops=1)
               Index Cond: (((subdivision)::text = 'East Uttar Pradesh'::text) AND (year = 1985))
   ->  GroupAggregate  (cost=0.29..51.59 rows=12 width=47) (actual time=22.786..22.787 rows=1 loops=1)
         Group Key: average_temperature.state, average_temperature.year
         ->  Index Scan using average_temperature_pkey on average_temperature  (cost=0.29..51.35 rows=12 width=21) (actual time=6.562..22.746 rows=12 loops=1)
               Index Cond: (((state)::text = 'Uttar Pradesh'::text) AND (year = 1985))
 Planning Time: 0.591 ms
 Execution Time: 32.429 ms
(13 rows)
```

After Indexing:

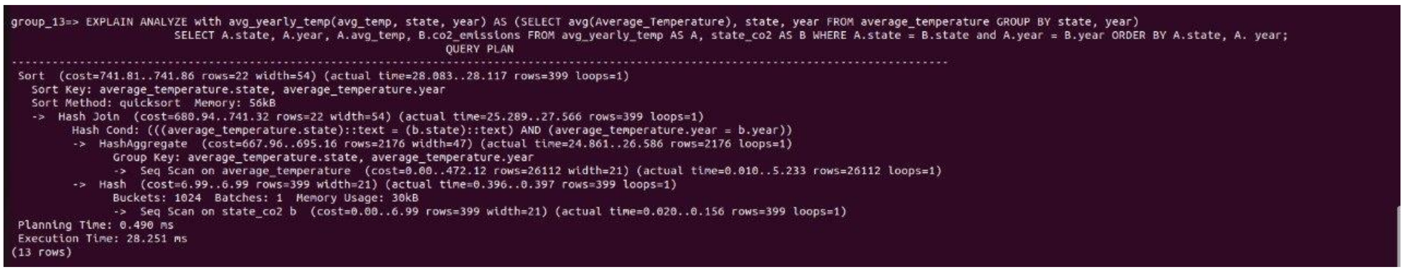
No indexes are used.



```
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=833.77..958.47 rows=1 width=243) (actual time=29.828..32.564 rows=1 loops=1)
   CTE rainfall_state
     ->  Hash Join  (cost=1.90..125.61 rows=2556 width=104) (actual time=0.047..1.905 rows=2556 loops=1)
           Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
           ->  Seq Scan on rainfall b  (cost=0.00..88.56 rows=2556 width=26) (actual time=0.006..0.471 rows=2556 loops=1)
           ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.020..0.020 rows=40 loops=1)
                 ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.004..0.008 rows=40 loops=1)
   CTE avg_yearly_temp
     ->  HashAggregate  (cost=680.96..708.16 rows=2176 width=23) (actual time=27.017..28.331 rows=2176 loops=1)
           ->  Seq Scan on average_temperature  (cost=0.00..485.12 rows=26112 width=23) (actual time=0.006..3.167 rows=26112 loops=1)
   ->  CTE Scan on avg_yearly_temp c  (cost=0.00..54.40 rows=1 width=114) (actual time=28.726..29.258 rows=1 loops=1)
         Filter: (((state)::text = 'Uttar Pradesh'::text) AND (year = 1985))
   ->  CTE Scan on rainfall_state d  (cost=0.00..70.29 rows=1 width=211) (actual time=1.100..3.304 rows=1 loops=1)
         Filter: (((d.state)::text = 'Uttar Pradesh'::text) AND ((d.subdivision)::text = 'East Uttar Pradesh'::text) AND (d.year = 1985))
 Total runtime: 32.729 ms
(15 rows)
```

Query 6:

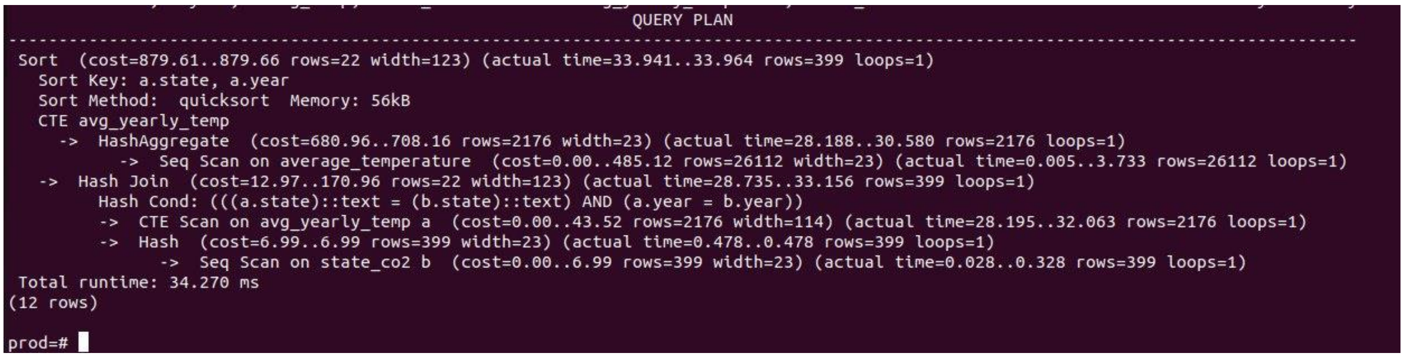
```
with avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state,
year FROM average_temperature GROUP BY state, year)
SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A, state_co2
AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;
```



```
group_13=> EXPLAIN ANALYZE with avg_yearly_temp(avg_temp, state, year) AS (SELECT avg(Average_Temperature), state, year FROM average_temperature GROUP BY state, year)
                        SELECT A.state, A.year, A.avg_temp, B.co2_emissions FROM avg_yearly_temp AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A. year;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=741.81..741.86 rows=22 width=54) (actual time=28.083..28.117 rows=399 loops=1)
   Sort Key: average_temperature.state, average_temperature.year
   Sort Method: quicksort  Memory: 56kB
   ->  Hash Join  (cost=680.94..741.32 rows=22 width=54) (actual time=25.289..27.566 rows=399 loops=1)
         Hash Cond: (((average_temperature.state)::text = (b.state)::text) AND (average_temperature.year = b.year))
         ->  HashAggregate  (cost=667.96..695.16 rows=2176 width=47) (actual time=24.861..26.586 rows=2176 loops=1)
               Group Key: average_temperature.state, average_temperature.year
               ->  Seq Scan on average_temperature  (cost=0.00..472.12 rows=26112 width=21) (actual time=0.010..5.233 rows=26112 loops=1)
         ->  Hash  (cost=6.99..6.99 rows=399 width=21) (actual time=0.396..0.397 rows=399 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 30kB
               ->  Seq Scan on state_co2 b  (cost=0.00..6.99 rows=399 width=21) (actual time=0.020..0.156 rows=399 loops=1)
 Planning Time: 0.490 ms
 Execution Time: 28.251 ms
(13 rows)
```

After Indexing:

No indexes are used.

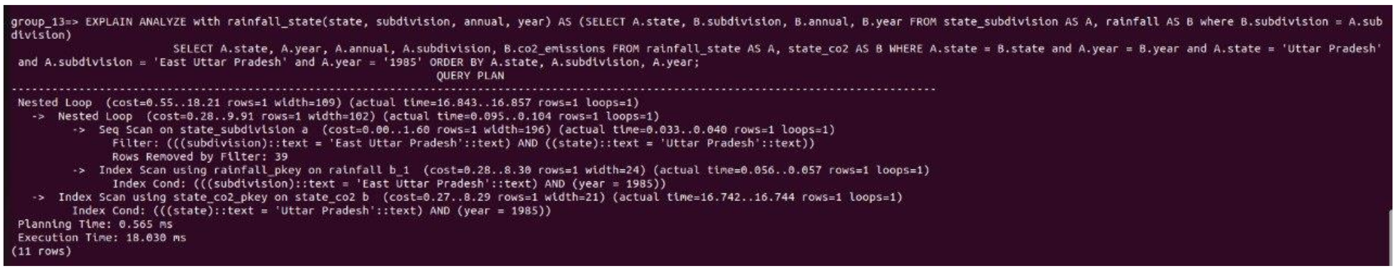


```
                                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=879.61..879.66 rows=22 width=123) (actual time=33.941..33.964 rows=399 loops=1)
   Sort Key: a.state, a.year
   Sort Method:  quicksort  Memory: 56kB
   CTE avg_yearly_temp
     ->  HashAggregate  (cost=680.96..708.16 rows=2176 width=23) (actual time=28.188..30.580 rows=2176 loops=1)
           ->  Seq Scan on average_temperature  (cost=0.00..485.12 rows=26112 width=23) (actual time=0.005..3.733 rows=26112 loops=1)
   ->  Hash Join  (cost=12.97..170.96 rows=22 width=123) (actual time=28.735..33.156 rows=399 loops=1)
         Hash Cond: (((a.state)::text = (b.state)::text) AND (a.year = b.year))
         ->  CTE Scan on avg_yearly_temp a  (cost=0.00..43.52 rows=2176 width=114) (actual time=28.195..32.063 rows=2176 loops=1)
         ->  Hash  (cost=6.99..6.99 rows=399 width=23) (actual time=0.478..0.478 rows=399 loops=1)
               ->  Seq Scan on state_co2 b  (cost=0.00..6.99 rows=399 width=23) (actual time=0.028..0.328 rows=399 loops=1)
 Total runtime: 34.270 ms
(12 rows)

prod=# █
```

Query 7:

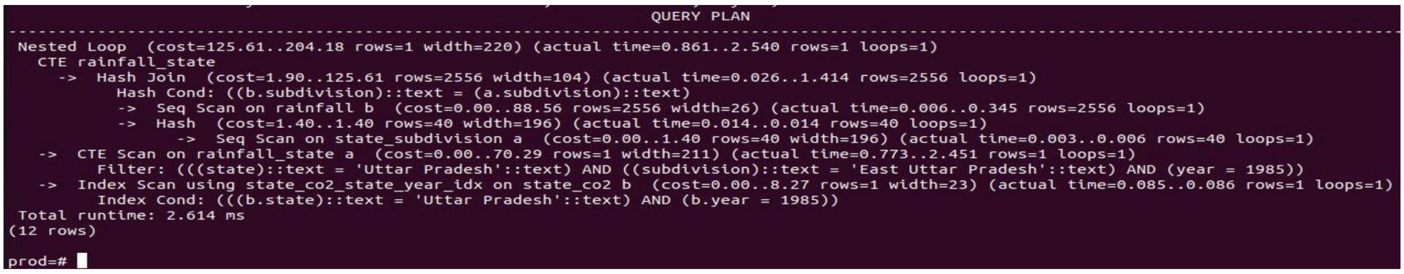
```sql
with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision,
B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision =
A.subdivision)
SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM rainfall_state AS A,
state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
and A.subdivision = 'East Uttar Pradesh' and A.year = '1985' ORDER BY A.state,
A.subdivision, A.year;
```



```
group_13=> EXPLAIN ANALYZE with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision = A.sub
division)
                        SELECT A.state, A.year, A.annual, A.subdivision, B.co2_emissions FROM rainfall_state AS A, state_co2 AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Uttar Pradesh'
 and A.subdivision = 'East Uttar Pradesh' and A.year = '1985' ORDER BY A.state, A.subdivision, A.year;
                                                                   QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.55..18.21 rows=1 width=109) (actual time=16.843..16.857 rows=1 loops=1)
   ->  Nested Loop  (cost=0.28..9.91 rows=1 width=102) (actual time=0.095..0.104 rows=1 loops=1)
         ->  Seq Scan on state_subdivision a  (cost=0.00..1.60 rows=1 width=196) (actual time=0.033..0.040 rows=1 loops=1)
               Filter: (((subdivision)::text = 'East Uttar Pradesh'::text) AND ((state)::text = 'Uttar Pradesh'::text))
               Rows Removed by Filter: 39
         ->  Index Scan using rainfall_pkey on rainfall b_1  (cost=0.28..8.30 rows=1 width=24) (actual time=0.056..0.057 rows=1 loops=1)
               Index Cond: (((subdivision)::text = 'East Uttar Pradesh'::text) AND (year = 1985))
   ->  Index Scan using state_co2_pkey on state_co2 b  (cost=0.27..8.29 rows=1 width=21) (actual time=16.742..16.744 rows=1 loops=1)
         Index Cond: (((state)::text = 'Uttar Pradesh'::text) AND (year = 1985))
 Planning Time: 0.565 ms
 Execution Time: 18.030 ms
(11 rows)
```

After Indexing:

Composite index on state and year attributes of state_co2 table is used by the query and the speedup is very apparent.

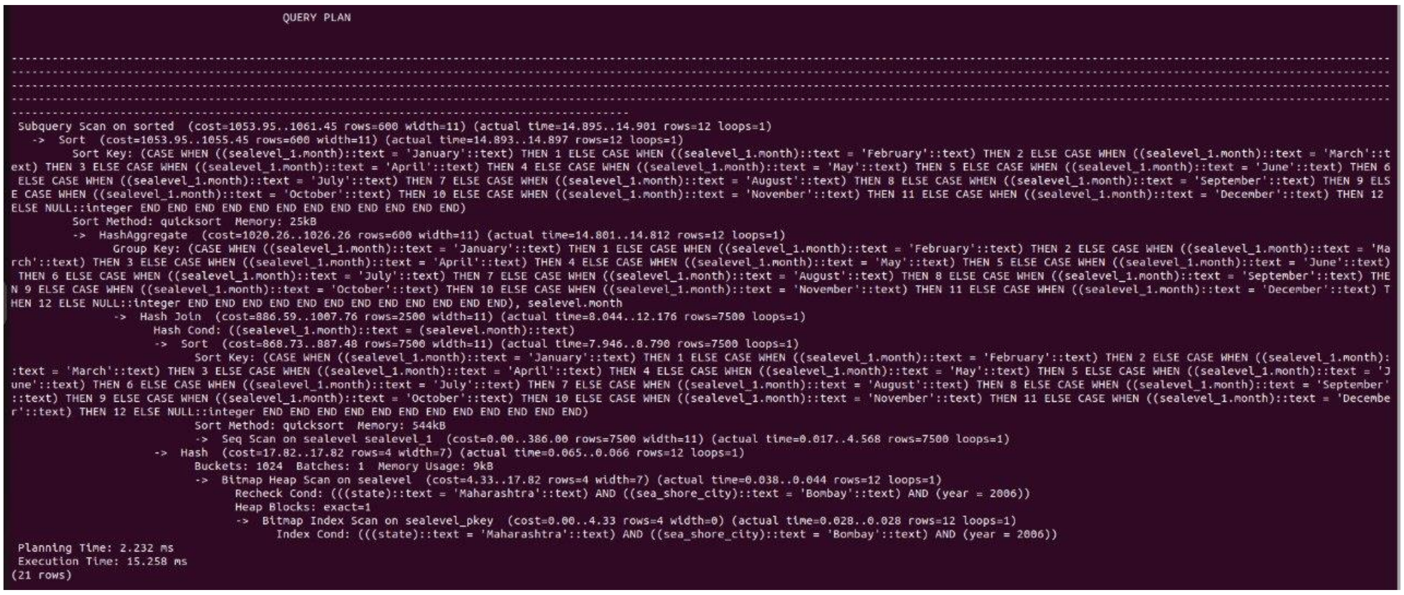


```
                                                                    QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=125.61..204.18 rows=1 width=220) (actual time=0.861..2.540 rows=1 loops=1)
   CTE rainfall_state
     ->  Hash Join  (cost=1.90..125.61 rows=2556 width=104) (actual time=0.026..1.414 rows=2556 loops=1)
           Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
           ->  Seq Scan on rainfall b  (cost=0.00..88.56 rows=2556 width=26) (actual time=0.006..0.345 rows=2556 loops=1)
           ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.014..0.014 rows=40 loops=1)
                 ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.003..0.006 rows=40 loops=1)
   ->  CTE Scan on rainfall_state a  (cost=0.00..70.29 rows=1 width=211) (actual time=0.773..2.451 rows=1 loops=1)
         Filter: (((state)::text = 'Uttar Pradesh'::text) AND ((subdivision)::text = 'East Uttar Pradesh'::text) AND (year = 1985))
   ->  Index Scan using state_co2_state_year_idx on state_co2 b  (cost=0.00..8.27 rows=1 width=23) (actual time=0.085..0.086 rows=1 loops=1)
         Index Cond: (((b.state)::text = 'Uttar Pradesh'::text) AND (b.year = 1985))
 Total runtime: 2.614 ms
(12 rows)

prod=# █
```

Query 8:

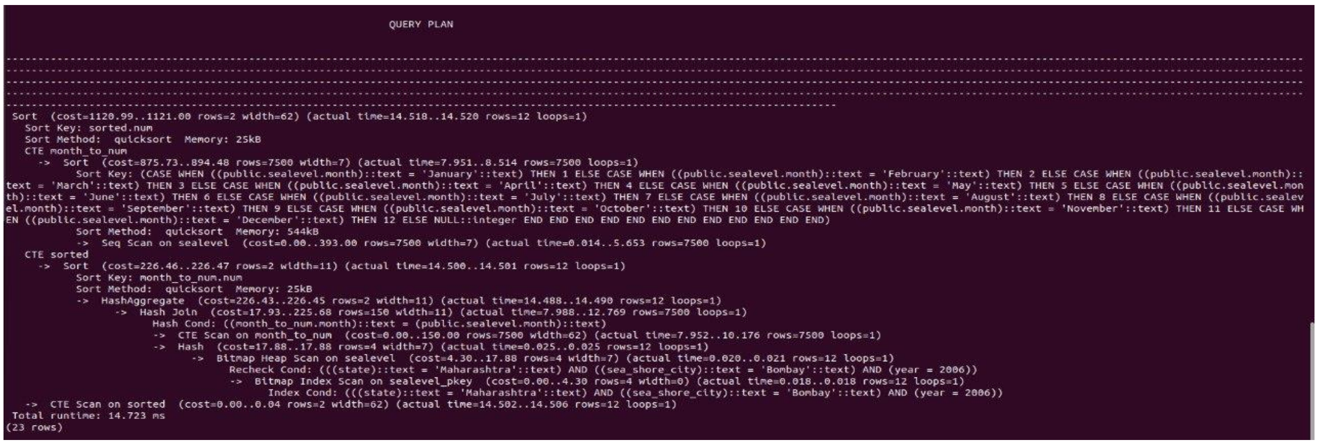
```sql
with month_to_num(month, num) AS (SELECT month, case when month='January' then 1 else case
when month = 'February' then 2 else case when month = 'March' then 3 else case when month
= 'April' then 4 else case when month = 'May' then 5 else case when month = 'June' then 6
else case when month = 'July' then 7 else case when month = 'August' then 8 else case when
month = 'September' then 9 else case when month = 'October' then 10 else case when month =
'November' then 11 else case when month = 'December' then 12 end end end end end end end
end end end end end AS num FROM sealevel ORDER BY num),
sorted(month) AS (SELECT DISTINCT sealevel.month, num FROM sealevel, month_to_num WHERE
state = 'Maharashtra' and sea_shore_city = 'Bombay' and year = '2006' and sealevel.month =
month_to_num.month ORDER BY month_to_num.num)
SELECT month FROM sorted ORDER BY num;
```



```
                                      QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
 Subquery Scan on sorted  (cost=1053.95..1061.45 rows=600 width=11) (actual time=14.895..14.901 rows=12 loops=1)
   ->  Sort  (cost=1053.95..1055.45 rows=600 width=11) (actual time=14.893..14.897 rows=12 loops=1)
         Sort Key: (CASE WHEN ((sealevel_1.month)::text = 'January'::text) THEN 1 ELSE CASE WHEN ((sealevel_1.month)::text = 'February'::text) THEN 2 ELSE CASE WHEN ((sealevel_1.month)::text = 'March'::t
ext) THEN 3 ELSE CASE WHEN ((sealevel_1.month)::text = 'April'::text) THEN 4 ELSE CASE WHEN ((sealevel_1.month)::text = 'May'::text) THEN 5 ELSE CASE WHEN ((sealevel_1.month)::text = 'June'::text) THEN 6
 ELSE CASE WHEN ((sealevel_1.month)::text = 'July'::text) THEN 7 ELSE CASE WHEN ((sealevel_1.month)::text = 'August'::text) THEN 8 ELSE CASE WHEN ((sealevel_1.month)::text = 'September'::text) THEN 9 ELS
E CASE WHEN ((sealevel_1.month)::text = 'October'::text) THEN 10 ELSE CASE WHEN ((sealevel_1.month)::text = 'November'::text) THEN 11 ELSE CASE WHEN ((sealevel_1.month)::text = 'December'::text) THEN 12
ELSE NULL::integer END END END END END END END END END END END END)
         Sort Method: quicksort  Memory: 25kB
         ->  HashAggregate  (cost=1020.26..1026.26 rows=600 width=11) (actual time=14.801..14.812 rows=12 loops=1)
               Group Key: (CASE WHEN ((sealevel_1.month)::text = 'January'::text) THEN 1 ELSE CASE WHEN ((sealevel_1.month)::text = 'February'::text) THEN 2 ELSE CASE WHEN ((sealevel_1.month)::text = 'Ma
rch'::text) THEN 3 ELSE CASE WHEN ((sealevel_1.month)::text = 'April'::text) THEN 4 ELSE CASE WHEN ((sealevel_1.month)::text = 'May'::text) THEN 5 ELSE CASE WHEN ((sealevel_1.month)::text = 'June'::text)
 THEN 6 ELSE CASE WHEN ((sealevel_1.month)::text = 'July'::text) THEN 7 ELSE CASE WHEN ((sealevel_1.month)::text = 'August'::text) THEN 8 ELSE CASE WHEN ((sealevel_1.month)::text = 'September'::text) THE
N 9 ELSE CASE WHEN ((sealevel_1.month)::text = 'October'::text) THEN 10 ELSE CASE WHEN ((sealevel_1.month)::text = 'November'::text) THEN 11 ELSE CASE WHEN ((sealevel_1.month)::text = 'December'::text) T
HEN 12 ELSE NULL::integer END END END END END END END END END END END END), sealevel.month
               ->  Hash Join  (cost=886.59..1007.76 rows=2500 width=11) (actual time=8.044..12.176 rows=7500 loops=1)
                     Hash Cond: ((sealevel_1.month)::text = (sealevel.month)::text)
                     ->  Sort  (cost=868.73..887.48 rows=7500 width=11) (actual time=7.946..8.790 rows=7500 loops=1)
                           Sort Key: (CASE WHEN ((sealevel_1.month)::text = 'January'::text) THEN 1 ELSE CASE WHEN ((sealevel_1.month)::text = 'February'::text) THEN 2 ELSE CASE WHEN ((sealevel_1.month):
:text = 'March'::text) THEN 3 ELSE CASE WHEN ((sealevel_1.month)::text = 'April'::text) THEN 4 ELSE CASE WHEN ((sealevel_1.month)::text = 'May'::text) THEN 5 ELSE CASE WHEN ((sealevel_1.month)::text = 'J
une'::text) THEN 6 ELSE CASE WHEN ((sealevel_1.month)::text = 'July'::text) THEN 7 ELSE CASE WHEN ((sealevel_1.month)::text = 'August'::text) THEN 8 ELSE CASE WHEN ((sealevel_1.month)::text = 'September'
::text) THEN 9 ELSE CASE WHEN ((sealevel_1.month)::text = 'October'::text) THEN 10 ELSE CASE WHEN ((sealevel_1.month)::text = 'November'::text) THEN 11 ELSE CASE WHEN ((sealevel_1.month)::text = 'Decembe
r'::text) THEN 12 ELSE NULL::integer END END END END END END END END END END END END)
                           Sort Method: quicksort  Memory: 544kB
                           ->  Seq Scan on sealevel sealevel_1  (cost=0.00..386.00 rows=7500 width=11) (actual time=0.017..4.568 rows=7500 loops=1)
                     ->  Hash  (cost=17.82..17.82 rows=4 width=7) (actual time=0.065..0.066 rows=12 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 9kB
                           ->  Bitmap Heap Scan on sealevel  (cost=4.33..17.82 rows=4 width=7) (actual time=0.038..0.044 rows=12 loops=1)
                                 Recheck Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shore_city)::text = 'Bombay'::text) AND (year = 2006))
                                 Heap Blocks: exact=1
                                 ->  Bitmap Index Scan on sealevel_pkey  (cost=0.00..4.33 rows=4 width=0) (actual time=0.028..0.028 rows=12 loops=1)
                                       Index Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shore_city)::text = 'Bombay'::text) AND (year = 2006))
 Planning Time: 2.232 ms
 Execution Time: 15.258 ms
(21 rows)
```

After Indexing:

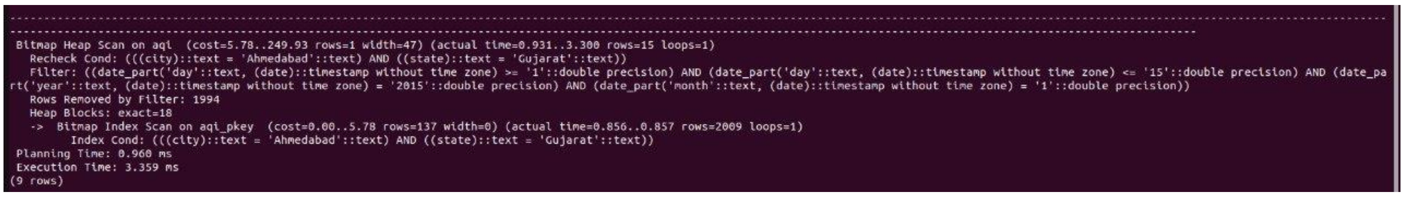
The implicit primary key index on relation sealevel is used by the query.



```
                                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=1120.99..1121.00 rows=2 width=62) (actual time=14.518..14.520 rows=12 loops=1)
   Sort Key: sorted.num
   Sort Method:  quicksort  Memory: 25kB
   CTE month_to_num
     ->  Sort  (cost=875.73..894.48 rows=7500 width=7) (actual time=7.951..8.514 rows=7500 loops=1)
           Sort Key: (CASE WHEN ((public.sealevel.month)::text = 'January'::text) THEN 1 ELSE CASE WHEN ((public.sealevel.month)::text = 'February'::text) THEN 2 ELSE CASE WHEN ((public.sealevel.month)::
text = 'March'::text) THEN 3 ELSE CASE WHEN ((public.sealevel.month)::text = 'April'::text) THEN 4 ELSE CASE WHEN ((public.sealevel.month)::text = 'May'::text) THEN 5 ELSE CASE WHEN ((public.sealevel.mon
th)::text = 'June'::text) THEN 6 ELSE CASE WHEN ((public.sealevel.month)::text = 'July'::text) THEN 7 ELSE CASE WHEN ((public.sealevel.month)::text = 'August'::text) THEN 8 ELSE CASE WHEN ((public.sealev
el.month)::text = 'September'::text) THEN 9 ELSE CASE WHEN ((public.sealevel.month)::text = 'October'::text) THEN 10 ELSE CASE WHEN ((public.sealevel.month)::text = 'November'::text) THEN 11 ELSE CASE WH
EN ((public.sealevel.month)::text = 'December'::text) THEN 12 ELSE NULL::integer END END END END END END END END END END END END)
           Sort Method:  quicksort  Memory: 544kB
           ->  Seq Scan on sealevel  (cost=0.00..393.00 rows=7500 width=7) (actual time=0.014..5.653 rows=7500 loops=1)
   CTE sorted
     ->  Sort  (cost=226.46..226.47 rows=2 width=11) (actual time=14.500..14.501 rows=12 loops=1)
           Sort Key: month_to_num.num
           Sort Method:  quicksort  Memory: 25kB
           ->  HashAggregate  (cost=226.43..226.45 rows=2 width=11) (actual time=14.488..14.490 rows=12 loops=1)
                 ->  Hash Join  (cost=17.93..225.68 rows=150 width=11) (actual time=7.988..12.769 rows=7500 loops=1)
                       Hash Cond: ((month_to_num.month)::text = (public.sealevel.month)::text)
                       ->  CTE Scan on month_to_num  (cost=0.00..150.00 rows=7500 width=62) (actual time=7.952..10.176 rows=7500 loops=1)
                       ->  Hash  (cost=17.88..17.88 rows=4 width=7) (actual time=0.025..0.025 rows=12 loops=1)
                             ->  Bitmap Heap Scan on sealevel  (cost=4.30..17.88 rows=4 width=7) (actual time=0.020..0.021 rows=12 loops=1)
                                   Recheck Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shore_city)::text = 'Bombay'::text) AND (year = 2006))
                                   ->  Bitmap Index Scan on sealevel_pkey  (cost=0.00..4.30 rows=4 width=0) (actual time=0.018..0.018 rows=12 loops=1)
                                         Index Cond: (((state)::text = 'Maharashtra'::text) AND ((sea_shore_city)::text = 'Bombay'::text) AND (year = 2006))
   ->  CTE Scan on sorted  (cost=0.00..0.04 rows=2 width=62) (actual time=14.502..14.506 rows=12 loops=1)
 Total runtime: 14.723 ms
(23 rows)
```

Query 9:

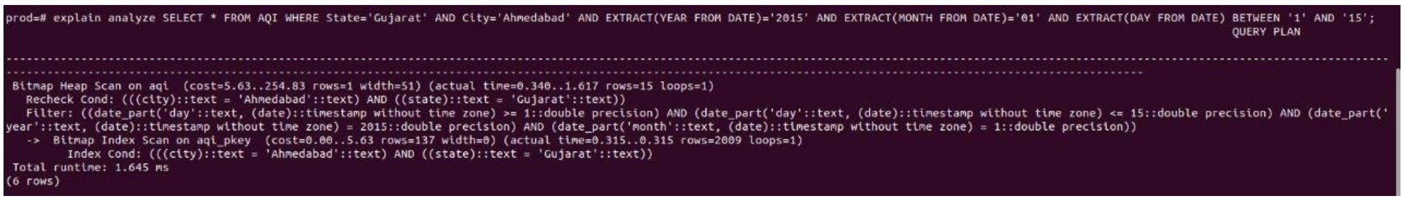
```
SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM
DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE) BETWEEN '1' AND
'15';
```



```
------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on aqi  (cost=5.78..249.93 rows=1 width=47) (actual time=0.931..3.300 rows=15 loops=1)
   Recheck Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
   Filter: ((date_part('day'::text, (date)::timestamp without time zone) >= '1'::double precision) AND (date_part('day'::text, (date)::timestamp without time zone) <= '15'::double precision) AND (date_pa
rt('year'::text, (date)::timestamp without time zone) = '2015'::double precision) AND (date_part('month'::text, (date)::timestamp without time zone) = '1'::double precision))
   Rows Removed by Filter: 1994
   Heap Blocks: exact=18
   ->  Bitmap Index Scan on aqi_pkey  (cost=0.00..5.78 rows=137 width=0) (actual time=0.856..0.857 rows=2009 loops=1)
         Index Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
 Planning Time: 0.960 ms
 Execution Time: 3.359 ms
(9 rows)
```

After Indexing:

The implicit primary key index is used by the query.

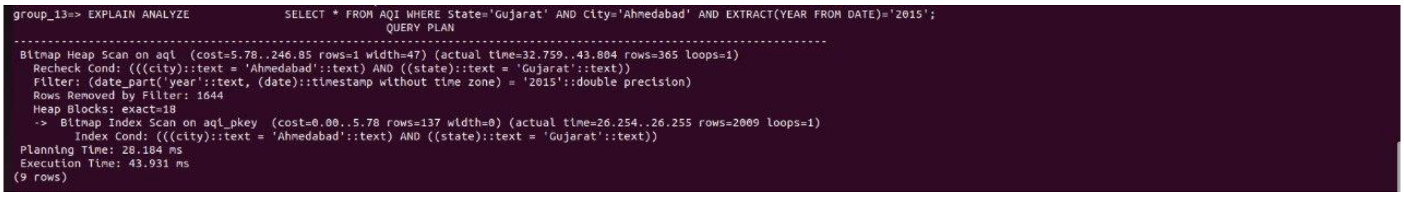


```
prod=# explain analyze SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE) BETWEEN '1' AND '15';
                                                                                                                                                                                    QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on aqi  (cost=5.63..254.83 rows=1 width=51) (actual time=0.340..1.617 rows=15 loops=1)
   Recheck Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
   Filter: ((date_part('day'::text, (date)::timestamp without time zone) >= 1::double precision) AND (date_part('day'::text, (date)::timestamp without time zone) <= 15::double precision) AND (date_part('
year'::text, (date)::timestamp without time zone) = 2015::double precision) AND (date_part('month'::text, (date)::timestamp without time zone) = 1::double precision))
   ->  Bitmap Index Scan on aqi_pkey  (cost=0.00..5.63 rows=137 width=0) (actual time=0.315..0.315 rows=2009 loops=1)
         Index Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
 Total runtime: 1.645 ms
(6 rows)
```

Query 10:

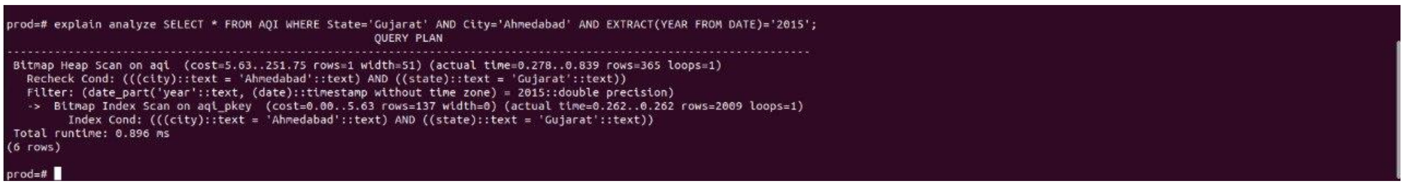
```sql
SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM
DATE)='2015';
```



```
group_13=> EXPLAIN ANALYZE             SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015';
                                                QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on aqi  (cost=5.78..246.85 rows=1 width=47) (actual time=32.759..43.804 rows=365 loops=1)
   Recheck Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
   Filter: (date_part('year'::text, (date)::timestamp without time zone) = '2015'::double precision)
   Rows Removed by Filter: 1644
   Heap Blocks: exact=18
   ->  Bitmap Index Scan on aqi_pkey  (cost=0.00..5.78 rows=137 width=0) (actual time=26.254..26.255 rows=2009 loops=1)
         Index Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
 Planning Time: 28.184 ms
 Execution Time: 43.931 ms
(9 rows)
```

After Indexing:

Even though the query plan is exactly the same, the difference in execution timing is huge!

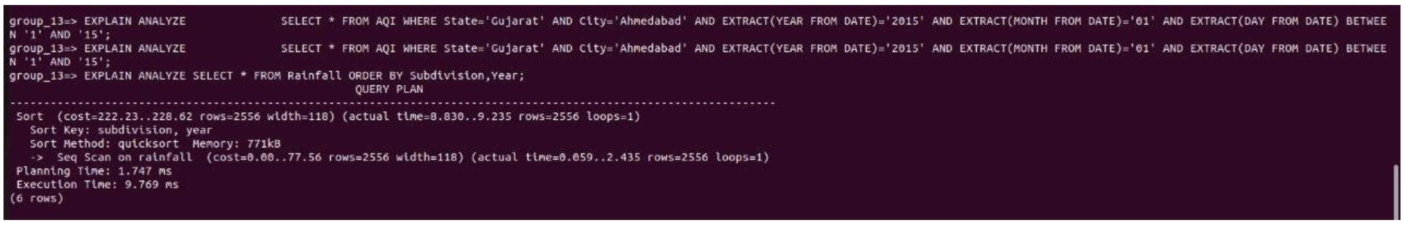


```
prod=# explain analyze SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015';
                                               QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on aqi  (cost=5.63..251.75 rows=1 width=51) (actual time=0.278..0.839 rows=365 loops=1)
   Recheck Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
   Filter: (date_part('year'::text, (date)::timestamp without time zone) = 2015::double precision)
   ->  Bitmap Index Scan on aqi_pkey  (cost=0.00..5.63 rows=137 width=0) (actual time=0.262..0.262 rows=2009 loops=1)
         Index Cond: (((city)::text = 'Ahmedabad'::text) AND ((state)::text = 'Gujarat'::text))
 Total runtime: 0.896 ms
(6 rows)

prod=# 
```

Query 11:

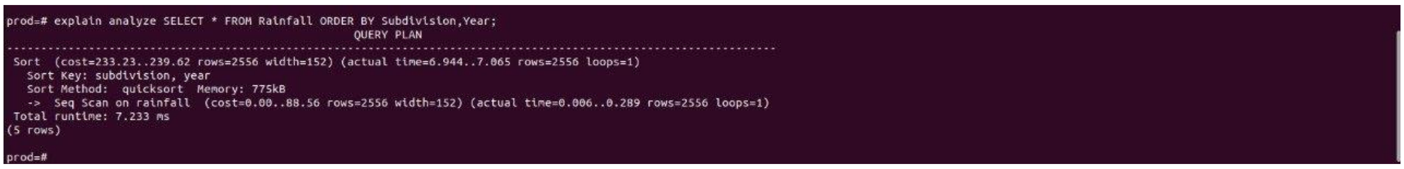
```sql
SELECT * FROM Rainfall ORDER BY Subdivision,Year;
```



```
group_13=> EXPLAIN ANALYZE             SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE) BETWEE
N '1' AND '15';
group_13=> EXPLAIN ANALYZE             SELECT * FROM AQI WHERE State='Gujarat' AND City='Ahmedabad' AND EXTRACT(YEAR FROM DATE)='2015' AND EXTRACT(MONTH FROM DATE)='01' AND EXTRACT(DAY FROM DATE) BETWEE
N '1' AND '15';
group_13=> EXPLAIN ANALYZE SELECT * FROM Rainfall ORDER BY Subdivision,Year;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Sort  (cost=222.23..228.62 rows=2556 width=118) (actual time=8.830..9.235 rows=2556 loops=1)
   Sort Key: subdivision, year
   Sort Method: quicksort  Memory: 771kB
   ->  Seq Scan on rainfall  (cost=0.00..77.56 rows=2556 width=118) (actual time=0.059..2.435 rows=2556 loops=1)
 Planning Time: 1.747 ms
 Execution Time: 9.769 ms
(6 rows)
```

After Indexing:

No indexes are used.

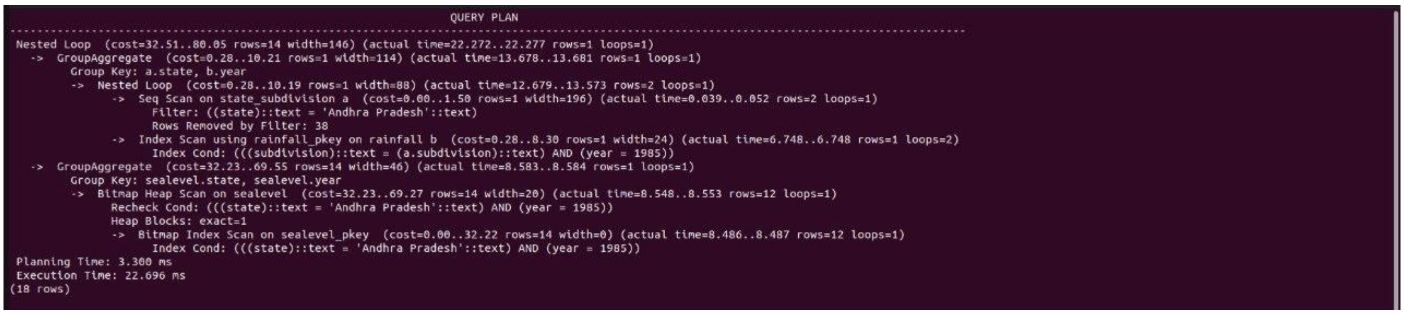


```
prod=# explain analyze SELECT * FROM Rainfall ORDER BY Subdivision,Year;
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
 Sort  (cost=233.23..239.62 rows=2556 width=152) (actual time=6.944..7.065 rows=2556 loops=1)
   Sort Key: subdivision, year
   Sort Method:  quicksort  Memory: 775kB
   ->  Seq Scan on rainfall  (cost=0.00..88.56 rows=2556 width=152) (actual time=0.006..0.289 rows=2556 loops=1)
 Total runtime: 7.233 ms
(5 rows)

prod=#
```

Query 12:

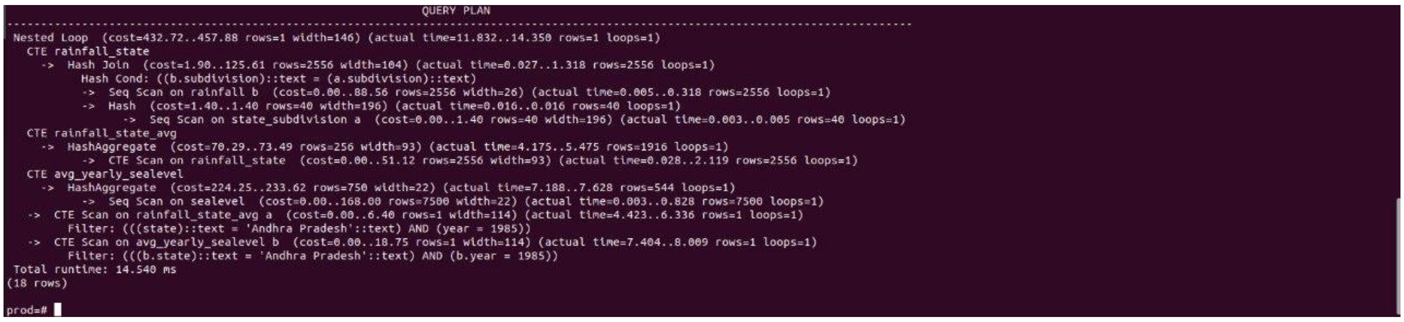
```
with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision,
B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision =
A.subdivision),
rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year FROM
rainfall_state GROUP BY state, year),
avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year, avg(monthly_msl) FROM
sealevel GROUP BY state, year)
SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A,
avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year and A.state = 'Andhra
Pradesh' and A.year = '1985' ORDER BY A.state, A.year;
```



```
                                                    QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=32.51..80.05 rows=14 width=146) (actual time=22.272..22.277 rows=1 loops=1)
   ->  GroupAggregate  (cost=0.28..10.21 rows=1 width=114) (actual time=13.678..13.681 rows=1 loops=1)
         Group Key: a.state, b.year
         ->  Nested Loop  (cost=0.28..10.19 rows=1 width=88) (actual time=12.679..13.573 rows=2 loops=1)
               ->  Seq Scan on state_subdivision a  (cost=0.00..1.50 rows=1 width=196) (actual time=0.039..0.052 rows=2 loops=1)
                     Filter: ((state)::text = 'Andhra Pradesh'::text)
                     Rows Removed by Filter: 38
               ->  Index Scan using rainfall_pkey on rainfall b  (cost=0.28..8.30 rows=1 width=24) (actual time=6.748..6.748 rows=1 loops=2)
                     Index Cond: (((subdivision)::text = (a.subdivision)::text) AND (year = 1985))
   ->  GroupAggregate  (cost=32.23..69.55 rows=14 width=46) (actual time=8.583..8.584 rows=1 loops=1)
         Group Key: sealevel.state, sealevel.year
         ->  Bitmap Heap Scan on sealevel  (cost=32.23..69.27 rows=14 width=20) (actual time=8.548..8.553 rows=12 loops=1)
               Recheck Cond: (((state)::text = 'Andhra Pradesh'::text) AND (year = 1985))
               Heap Blocks: exact=1
               ->  Bitmap Index Scan on sealevel_pkey  (cost=0.00..32.22 rows=14 width=0) (actual time=8.486..8.487 rows=12 loops=1)
                     Index Cond: (((state)::text = 'Andhra Pradesh'::text) AND (year = 1985))
 Planning Time: 3.300 ms
 Execution Time: 22.696 ms
(18 rows)
```

After Indexing:

No indexes are used.

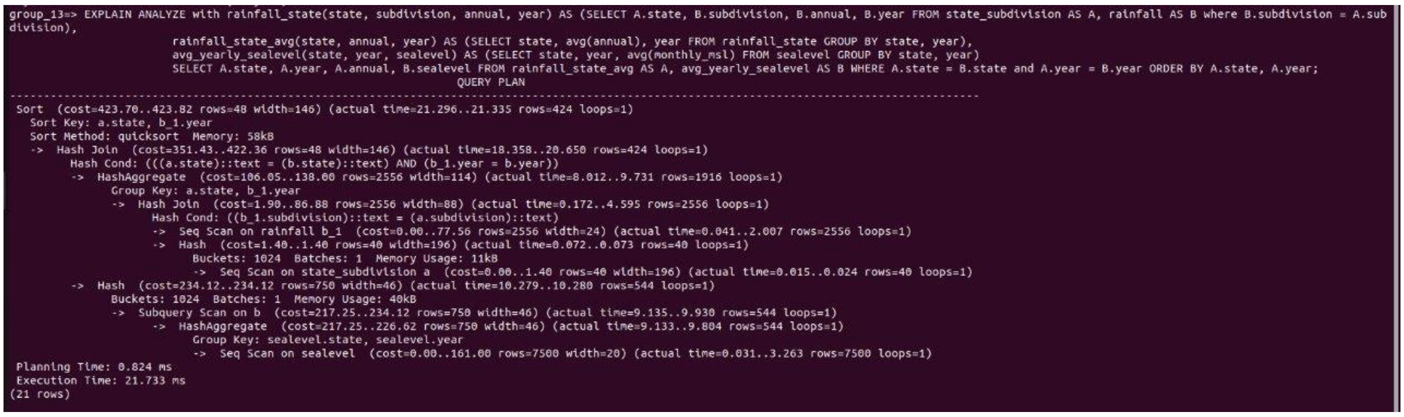


```
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=432.72..457.88 rows=1 width=146) (actual time=11.832..14.350 rows=1 loops=1)
   CTE rainfall_state
     ->  Hash Join  (cost=1.90..125.61 rows=2556 width=104) (actual time=0.027..1.318 rows=2556 loops=1)
           Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
           ->  Seq Scan on rainfall b  (cost=0.00..88.56 rows=2556 width=26) (actual time=0.005..0.318 rows=2556 loops=1)
           ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.016..0.016 rows=40 loops=1)
                 ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.003..0.005 rows=40 loops=1)
   CTE rainfall_state_avg
     ->  HashAggregate  (cost=70.29..73.49 rows=256 width=93) (actual time=4.175..5.475 rows=1916 loops=1)
           ->  CTE Scan on rainfall_state  (cost=0.00..51.12 rows=2556 width=93) (actual time=0.028..2.119 rows=2556 loops=1)
   CTE avg_yearly_sealevel
     ->  HashAggregate  (cost=224.25..233.62 rows=750 width=22) (actual time=7.188..7.628 rows=544 loops=1)
           ->  Seq Scan on sealevel  (cost=0.00..168.00 rows=7500 width=22) (actual time=0.003..0.828 rows=7500 loops=1)
   ->  CTE Scan on rainfall_state_avg a  (cost=0.00..6.40 rows=1 width=114) (actual time=4.423..6.336 rows=1 loops=1)
         Filter: (((state)::text = 'Andhra Pradesh'::text) AND (year = 1985))
   ->  CTE Scan on avg_yearly_sealevel b  (cost=0.00..18.75 rows=1 width=114) (actual time=7.404..8.009 rows=1 loops=1)
         Filter: (((b.state)::text = 'Andhra Pradesh'::text) AND (b.year = 1985))
 Total runtime: 14.540 ms
(18 rows)

prod=#
```

Query 13:

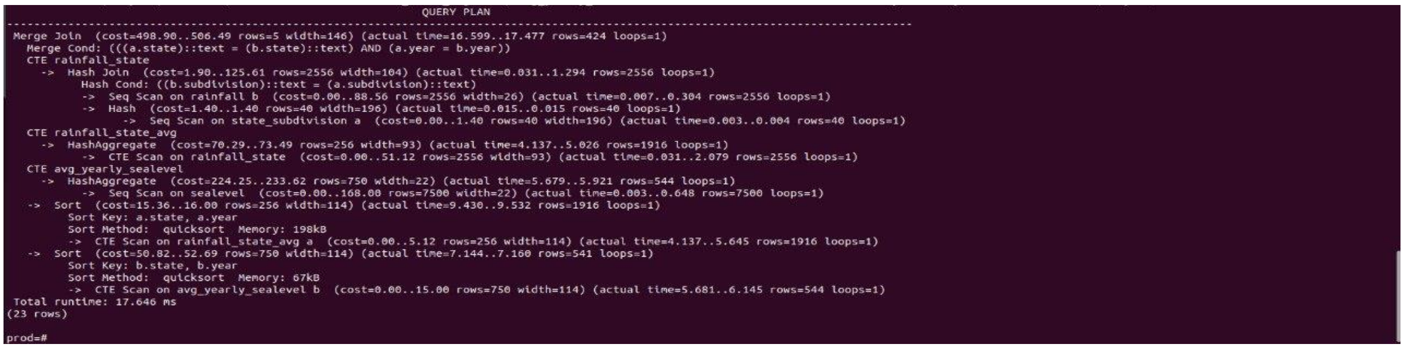
```
with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision,
B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision =
A.subdivision),
rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year FROM
rainfall_state GROUP BY state, year),
avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year, avg(monthly_msl) FROM
sealevel GROUP BY state, year)
SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A,
avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state,
A.year;
```



```
group_13=> EXPLAIN ANALYZE with rainfall_state(state, subdivision, annual, year) AS (SELECT A.state, B.subdivision, B.annual, B.year FROM state_subdivision AS A, rainfall AS B where B.subdivision = A.sub
division),
                        rainfall_state_avg(state, annual, year) AS (SELECT state, avg(annual), year FROM rainfall_state GROUP BY state, year),
                        avg_yearly_sealevel(state, year, sealevel) AS (SELECT state, year, avg(monthly_msl) FROM sealevel GROUP BY state, year)
                        SELECT A.state, A.year, A.annual, B.sealevel FROM rainfall_state_avg AS A, avg_yearly_sealevel AS B WHERE A.state = B.state and A.year = B.year ORDER BY A.state, A.year;
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=423.70..423.82 rows=48 width=146) (actual time=21.296..21.335 rows=424 loops=1)
   Sort Key: a.state, b_1.year
   Sort Method: quicksort  Memory: 58kB
   ->  Hash Join  (cost=351.43..422.36 rows=48 width=146) (actual time=18.358..20.650 rows=424 loops=1)
         Hash Cond: (((a.state)::text = (b.state)::text) AND (b_1.year = b.year))
         ->  HashAggregate  (cost=106.05..138.00 rows=2556 width=114) (actual time=8.012..9.731 rows=1916 loops=1)
               Group Key: a.state, b_1.year
               ->  Hash Join  (cost=1.90..86.88 rows=2556 width=88) (actual time=0.172..4.595 rows=2556 loops=1)
                     Hash Cond: ((b_1.subdivision)::text = (a.subdivision)::text)
                     ->  Seq Scan on rainfall b_1  (cost=0.00..77.56 rows=2556 width=24) (actual time=0.041..2.007 rows=2556 loops=1)
                     ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.072..0.073 rows=40 loops=1)
                           Buckets: 1024  Batches: 1  Memory Usage: 11kB
                           ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.015..0.024 rows=40 loops=1)
         ->  Hash  (cost=234.12..234.12 rows=750 width=46) (actual time=10.279..10.280 rows=544 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 40kB
               ->  Subquery Scan on b  (cost=217.25..234.12 rows=750 width=46) (actual time=9.135..9.930 rows=544 loops=1)
                     ->  HashAggregate  (cost=217.25..226.62 rows=750 width=46) (actual time=9.133..9.804 rows=544 loops=1)
                           Group Key: sealevel.state, sealevel.year
                           ->  Seq Scan on sealevel  (cost=0.00..161.00 rows=7500 width=20) (actual time=0.031..3.263 rows=7500 loops=1)
 Planning Time: 0.824 ms
 Execution Time: 21.733 ms
(21 rows)
```

After Indexing:

No indexes are used.

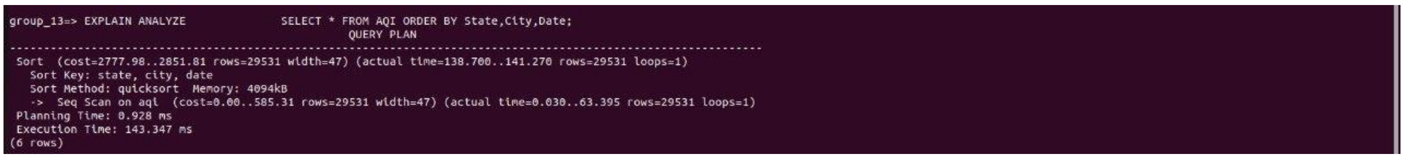


```
                                                        QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Merge Join  (cost=498.90..506.49 rows=5 width=146) (actual time=16.599..17.477 rows=424 loops=1)
   Merge Cond: (((a.state)::text = (b.state)::text) AND (a.year = b.year))
   CTE rainfall_state
     ->  Hash Join  (cost=1.90..125.61 rows=2556 width=104) (actual time=0.031..1.294 rows=2556 loops=1)
           Hash Cond: ((b.subdivision)::text = (a.subdivision)::text)
           ->  Seq Scan on rainfall b  (cost=0.00..88.56 rows=2556 width=26) (actual time=0.007..0.304 rows=2556 loops=1)
           ->  Hash  (cost=1.40..1.40 rows=40 width=196) (actual time=0.015..0.015 rows=40 loops=1)
                 ->  Seq Scan on state_subdivision a  (cost=0.00..1.40 rows=40 width=196) (actual time=0.003..0.004 rows=40 loops=1)
   CTE rainfall_state_avg
     ->  HashAggregate  (cost=70.29..73.49 rows=256 width=93) (actual time=4.137..5.026 rows=1916 loops=1)
           ->  CTE Scan on rainfall_state  (cost=0.00..51.12 rows=2556 width=93) (actual time=0.031..2.079 rows=2556 loops=1)
   CTE avg_yearly_sealevel
     ->  HashAggregate  (cost=224.25..233.62 rows=750 width=22) (actual time=5.679..5.921 rows=544 loops=1)
           ->  Seq Scan on sealevel  (cost=0.00..168.00 rows=7500 width=22) (actual time=0.003..0.648 rows=7500 loops=1)
   ->  Sort  (cost=15.36..16.00 rows=256 width=114) (actual time=9.430..9.532 rows=1916 loops=1)
         Sort Key: a.state, a.year
         Sort Method:  quicksort  Memory: 198kB
         ->  CTE Scan on rainfall_state_avg a  (cost=0.00..5.12 rows=256 width=114) (actual time=4.137..5.645 rows=1916 loops=1)
   ->  Sort  (cost=50.82..52.69 rows=750 width=114) (actual time=7.144..7.160 rows=541 loops=1)
         Sort Key: b.state, b.year
         Sort Method:  quicksort  Memory: 67kB
         ->  CTE Scan on avg_yearly_sealevel b  (cost=0.00..15.00 rows=750 width=114) (actual time=5.681..6.145 rows=544 loops=1)
 Total runtime: 17.646 ms
(23 rows)

prod=#
```

Query 14:

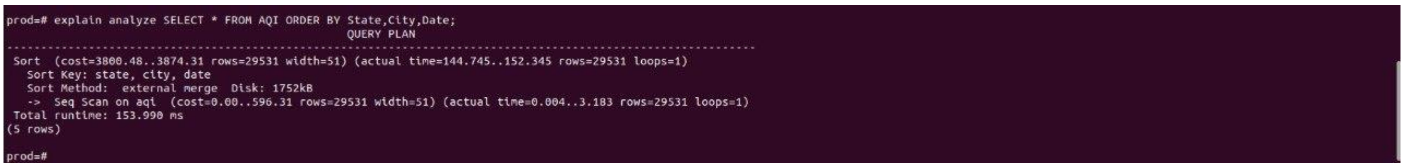
```sql
SELECT * FROM AQI ORDER BY State,City,Date;
```



```
group_13=> EXPLAIN ANALYZE              SELECT * FROM AQI ORDER BY State,City,Date;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Sort  (cost=2777.98..2851.81 rows=29531 width=47) (actual time=138.700..141.270 rows=29531 loops=1)
   Sort Key: state, city, date
   Sort Method: quicksort  Memory: 4094kB
   ->  Seq Scan on aqi  (cost=0.00..585.31 rows=29531 width=47) (actual time=0.030..63.395 rows=29531 loops=1)
 Planning Time: 0.928 ms
 Execution Time: 143.347 ms
(6 rows)
```

After Indexing:

No indexes are used.



```
prod=# explain analyze SELECT * FROM AQI ORDER BY State,City,Date;
                                                  QUERY PLAN
------------------------------------------------------------------------------------------------------------
 Sort  (cost=3800.48..3874.31 rows=29531 width=51) (actual time=144.745..152.345 rows=29531 loops=1)
   Sort Key: state, city, date
   Sort Method:  external merge  Disk: 1752kB
   ->  Seq Scan on aqi  (cost=0.00..596.31 rows=29531 width=51) (actual time=0.004..3.183 rows=29531 loops=1)
 Total runtime: 153.990 ms
(5 rows)

prod=#
```