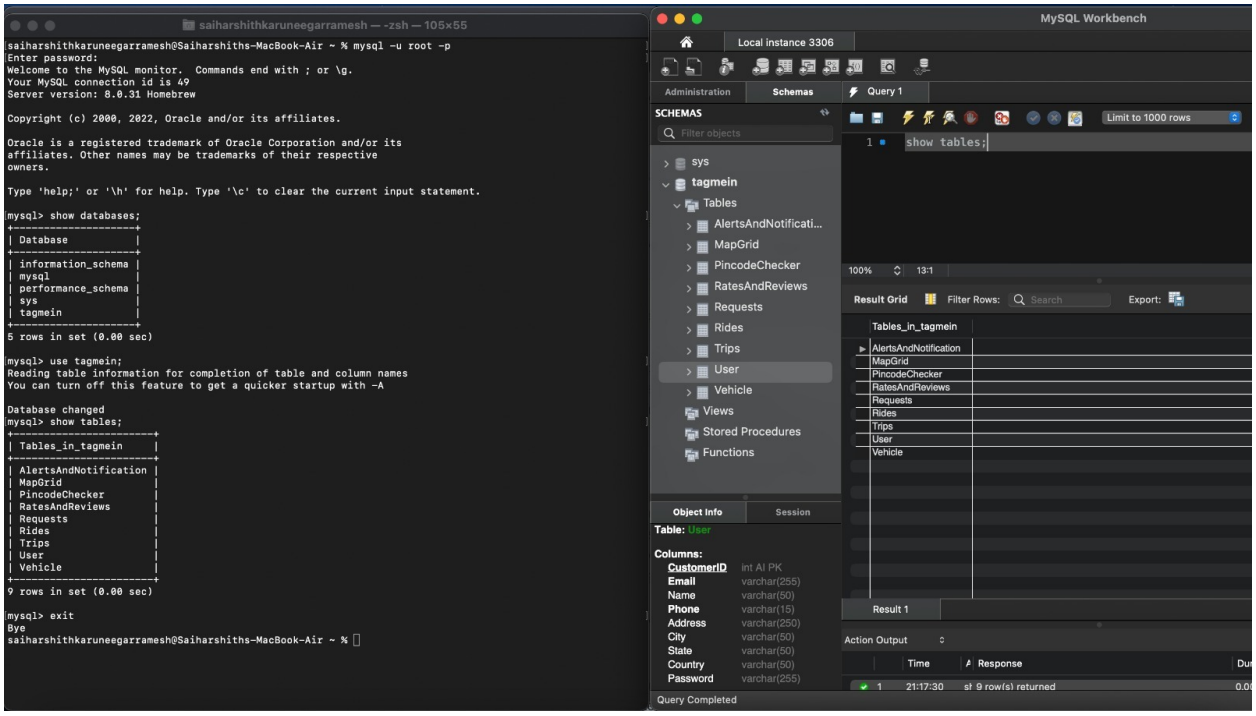
STAGE 3 - THE BOYS - TEAM 105

I. TABLES IMPLEMENTATIONS
USER, VEHICLE, TRIPS, MAP GRID, RIDES, RATES AND REVIEWS



II. Data Definition Language (DDL) Commands

TABLE 1 - User:

CREATE TABLE User (CustomerID INT PRIMARY KEY AUTO_INCREMENT, Email VARCHAR(255) UNIQUE NOT NULL, Name VARCHAR(50) NOT NULL, Phone VARCHAR(15) UNIQUE NOT NULL, Address VARCHAR(250) NOT NULL, City VARCHAR(50) NOT NULL, State VARCHAR(50) NOT NULL, Country VARCHAR(50) NOT NULL, Password VARCHAR(255) NOT NULL)

TABLE 2 - Vehicle:

CREATE TABLE Vehicle (
VehicleID INT PRIMARY KEY AUTO_INCREMENT,
OwnerID INT,

```
CarName VARCHAR(30) NOT NULL,
CarType VARCHAR(30),
RegistrationNumber VARCHAR(40) UNIQUE NOT NULL,
LicenseNumber VARCHAR(30) UNIQUE NOT NULL,
Color VARCHAR(20) NOT NULL,
NoOfSeats INT NOT NULL,
CHECK (NoOfSeats <=10 ),
FOREIGN KEY (OwnerID) REFERENCES User(CustomerID) ON DELETE SET NULL
);
```

TABLE 3 - Trips:

```
CREATE TABLE Trips (
TripID INT PRIMARY KEY AUTO_INCREMENT,
VehicleID INT,
Date DATE NOT NULL,
Time TIMESTAMP NOT NULL,
Source VARCHAR(255) NOT NULL,
Destination VARCHAR(255) NOT NULL,
Seats INT NOT NULL,
Status VARCHAR(10) NOT NULL,
FOREIGN KEY(VehicleID) REFERENCES Vehicle(VehicleID) ON DELETE SET NULL
);
```

TABLE 4 - MapGrid:

```
CREATE Table MapGrid (
TripID INT PRIMARY KEY,
SourceLatitude REAL NOT NULL,
DestinationLatitude REAL NOT NULL,
SourceLongitude REAL NOT NULL,
DestinationLongitude REAL NOT NULL,
FOREIGN KEY(TripID) REFERENCES TRIPS(TripID) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

TABLE 5 - Rides:

```
CREATE TABLE Rides(
TripID INT,
RiderID INT,
```

```
PRIMARY KEY(TripID, RiderID),
FOREIGN KEY(RiderID) REFERENCES User(CustomerID) ON UPDATE CASCADE
ON DELETE CASCADE,
FOREIGN KEY(TripID) REFERENCES Trips(TripID) ON UPDATE CASCADE ON
DELETE CASCADE
);
```

TABLE 6 - RatesAndReviews:

```
CREATE Table RatesAndReviews(
CustomerID INT,
TripID INT,
Review VARCHAR(255) NOT NULL,
Rating REAL NOT NULL,
PRIMARY KEY(CustomerID, TripID),
FOREIGN KEY(CustomerID) REFERENCES User(CustomerID) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY(TripID) REFERENCES Trips(TripID) ON DELETE CASCADE
);
```

TABLE 7 - PincodeChecker

```
CREATE Table PincodeChecker(
Pincode INT PRIMARY KEY,
AreaName VARCHAR(100) NOT NULL
);
```

TABLE 8 - AlertsAndNotification

```
CREATE Table AlertsAndNotification(
AlertID INT PRIMARY KEY AUTO_INCREMENT,
CustomerId INT,
Message VARCHAR(255) NOT NULL,
Status VARCHAR(15) NOT NULL,
FOREIGN KEY(CustomerID) REFERENCES User(CustomerID) ON DELETE
CASCADE
);
```

TABLE 9 - Requests:

```
CREATE Table Requests (
RequestID INT PRIMARY KEY AUTO_INCREMENT,
RiderID INT,
Comment VARCHAR(255) NOT NULL,
Seats INT NOT NULL,
Status VARCHAR(50) NOT NULL,
FOREIGN KEY(RiderID) REFERENCES User(CustomerID) ON DELETE CASCADE,
CHECK(Seats <=10)
);
```

III. EXECUTION SCREENSHOTS

TABLE 1 - User

```
mysql> select * from User limit 5;
+------------+---------------------------+---------------------------+----------------+-----------------------------+-----------------+--------------+---------+----------------------------------
----+
| CustomerID | Email                     | Name                      | Phone          | Address                     | City            | State        | Country | Password
    |
+------------+---------------------------+---------------------------+----------------+-----------------------------+-----------------+--------------+---------+----------------------------------
----+
|          1 | o'reilly.wilma@hotmail.com | Weston Sawayn Sr.         | 839.566.1501x92 | 29950 Wisozk Hills Apt. 341  | Port Josefaview | Texas        | "USA"   | d44ca03ec1876fa47e64e300ec431f6e201ea
c8b |
|          2 | abel.hirthe@yahoo.com      | Ms. Luna Rutherford V     | +31(4)074382678 | 73114 Wendell Row Apt. 936   | Lake Neil       | Massachusetts | "USA"   | 22209dc67ce65bd6b3ef1eca4121583bab60f
2b8 |
|          3 | caitlyn64@yahoo.com        | Jairo Beier               | 564.612.8076x97 | 5506 Milford Divide          | Lake Cletafort  | Wisconsin    | "USA"   | 472babddf5e2e10817213dddaefaaf2387e4f
f25 |
|          4 | wehner.domenica@yahoo.com  | Prof. Greyson Jacobson III | (689)157-5100x3 | 6751 Schaefer Course Apt. 030 | West Tatum      | Kentucky     | "USA"   | d210a5474d82e2dc231fe8ade6e29929426a6
9f5 |
|          5 | olen28@hotmail.com         | Marianna Parker           | (070)819-0854  | 89319 Lexus Tunnel Suite 235 | South Kaylie    | Montana      | "USA"   | 2d63fce3b04646f9491aeec7597b23fd9abff
8ea |
+------------+---------------------------+---------------------------+----------------+-----------------------------+-----------------+--------------+---------+----------------------------------
----+
5 rows in set (0.00 sec)

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

TABLE 2 - Vehicle

```
mysql> select * from Vehicle limit 5;
+------------+----------+-----------+-------------+--------------------+----------------+-----------+------------+
| VehicleID | OwnerID | CarName   | CarType     | RegistrationNumber | LicenseNumber  | Color     | NoOfSeats |
+------------+----------+-----------+-------------+--------------------+----------------+-----------+------------+
|          1 |      550 |  "RAM"    |  "van"      | 417490mk           | s843-0070-5005 |  "Red"    |          4 |
|          2 |      699 |  "Toyota" |  "Hachback" | 606185mt           | j621-2695-3084 |  "Red"    |          3 |
|          3 |      598 |  "Dodge"  |  "Hachback" | 809870iz           | d572-0205-1689 | "Black"   |          2 |
|          4 |       34 |  "Nissan" |  "Other"    | 824529iq           | p948-0212-3947 | "Black"   |          2 |
|          5 |      585 | "other""  | "Suv"       | 512000fg           | x283-3347-2502 |  "Green"  |          5 |
+------------+----------+-----------+-------------+--------------------+----------------+-----------+------------+
5 rows in set (0.00 sec)

mysql> select count(*) from Vehicle;
+----------+
| count(*) |
+----------+
|     1500 |
+----------+
1 row in set (0.00 sec)
```

TABLE 3 - Trips

```
mysql> select * from Trips limit 5;
+--------+-----------+------------+---------------------+--------------------------+-------------------------------+-------+----------+
| TripID | VehicleID | Date       | Time                | Source                   | Destination                   | Seats | Status   |
+--------+-----------+------------+---------------------+--------------------------+-------------------------------+-------+----------+
|      1 |       865 | 2022-08-21 | 2022-08-21 18:02:38 | 2216 S Stonebrooke Ct Urbana | 3404 Boulder Ridge Dr Champaign |     3 | Past     |
|      2 |       821 | 2022-10-18 | 2022-10-18 12:45:12 | 2216 S Stonebrooke Ct Urbana | 3402 Boulder Ridge Dr Champaign |     4 | Upcoming |
|      3 |       405 | 2022-09-24 | 2022-09-24 18:31:19 | 2216 S Stonebrooke Ct Urbana | 2906 Greystone Pl Champaign     |     5 | Past     |
|      4 |        64 | 2022-08-16 | 2022-08-16 05:06:04 | 2216 S Stonebrooke Ct Urbana | 5006 EMMAS WAY Champaign        |     5 | Past     |
|      5 |       269 | 2022-07-16 | 2022-07-16 02:14:51 | 2216 S Stonebrooke Ct Urbana | 2401 CLAYTON Champaign          |     4 | Past     |
+--------+-----------+------------+---------------------+--------------------------+-------------------------------+-------+----------+
5 rows in set (0.00 sec)

mysql> select count(*) from Trips;
+----------+
| count(*) |
+----------+
|    14115 |
+----------+
1 row in set (0.00 sec)
```

TABLE 4 - RatesAndReviews

```
mysql> select * from RatesAndReviews limit 5;
+------------+--------+---------------------+--------+
| CustomerID | TripID | Review              | Rating |
+------------+--------+---------------------+--------+
|        500 |    159 | This Ride was Good |      5 |
|        500 |    246 | This Ride was Good |    1.5 |
|        500 |    571 | This Ride was Good |    2.5 |
|        500 |    574 | This Ride was Good |      2 |
|        500 |    660 | This Ride was Good |      0 |
+------------+--------+---------------------+--------+
5 rows in set (0.01 sec)

mysql> select count(*) from RatesAndReviews;
+----------+
| count(*) |
+----------+
|    57113 |
+----------+
1 row in set (0.01 sec)
```

TABLE 5 - MapGrid:

```
mysql> select * from MapGrid limit 5;
+--------+------------------+---------------------+------------------+----------------------+
| TripID | SourceLatitude   | DestinationLatitude | SourceLongitude  | DestinationLongitude |
+--------+------------------+---------------------+------------------+----------------------+
|      1 | 40.0890390566996 |    40.1321952231074 | -88.1638343471284 |     -88.303030555667 |
|      2 | 40.0890390566996 |    40.1322918363534 | -88.1638343471284 |    -88.3028926046963 |
|      3 | 40.0890390566996 |    40.0736758181979 | -88.1638343471284 |    -88.2946842814426 |
|      4 | 40.0890390566996 |    40.0895255118398 | -88.1638343471284 |    -88.3013026274598 |
|      5 | 40.0890390566996 |    40.1274226711593 | -88.1638343471284 |    -88.2868083248039 |
+--------+------------------+---------------------+------------------+----------------------+
5 rows in set (0.00 sec)

mysql> select count(*) from MapGrid;
+----------+
| count(*) |
+----------+
|    14115 |
+----------+
1 row in set (0.00 sec)
```

TABLE 6 - Rides:

```
mysql> select * from Rides limit 5;
+--------+---------+
| TripID | RiderID |
+--------+---------+
|    159 |     500 |
|    246 |     500 |
|    571 |     500 |
|    574 |     500 |
|    660 |     500 |
+--------+---------+
5 rows in set (0.00 sec)

mysql> select count(*) from Rides;
+----------+
| count(*) |
+----------+
|    57113 |
+----------+
1 row in set (0.01 sec)
```

IV. ADVANCED QUERIES

FIRST COMPLEX QUERY

1) Each owner can check the aggregated average rating of each of his vehicles. For Example: If an owner has three different cars like Audi, Toyota and Maruti. Owner can see the overall aggregate rating for each of his/her vehicles. In this way, the owner can improve service of a particular car based on the user reviews. [Combination of Join of multiple relations, subqueries and Aggregation via GROUP BY]

Query:

```
EXPLAIN ANALYZE
SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews
NATURAL JOIN
SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
(SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
Group BY VehicleID;
```

Output screenshot:

```
2  SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
3  (SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
4  (SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
5  Group BY VehicleID;
6
```

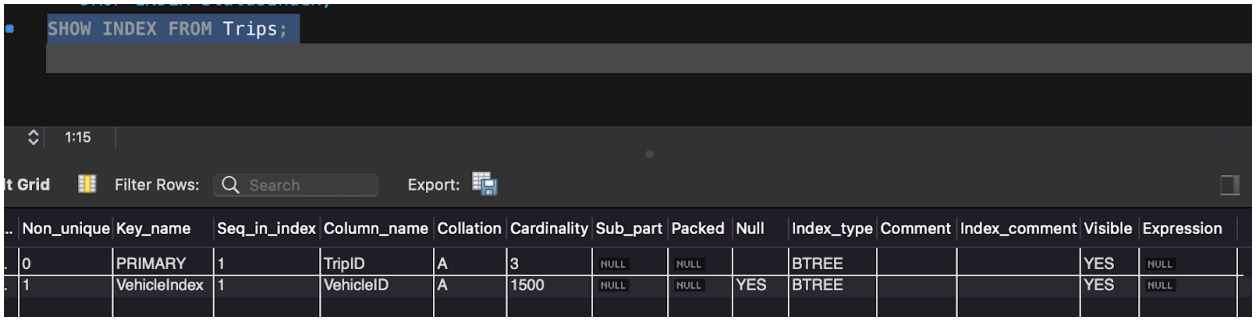
100% 1:6

Result Grid Filter Rows: Search Export:

VehicleID	AvgRating
409	2.5833333333333335
594	2.96875
712	1.9
957	2.71875

Performance:

Before Indexing: (with mysql's default indexing)

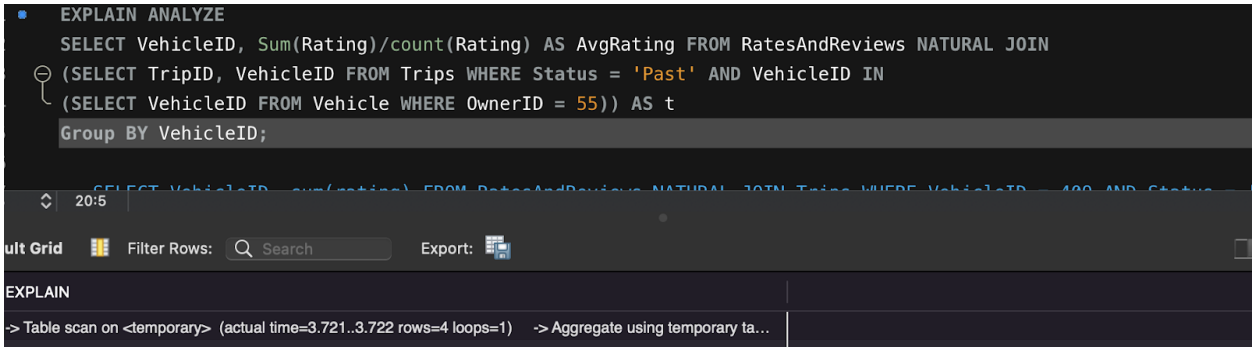


```
SHOW INDEX FROM Trips;
```

1:15

Filter Rows: Search Export:

Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
0	PRIMARY	1	TripID	A	3	NULL	NULL		BTREE			YES	NULL
1	VehicleIndex	1	VehicleID	A	1500	NULL	NULL	YES	BTREE			YES	NULL



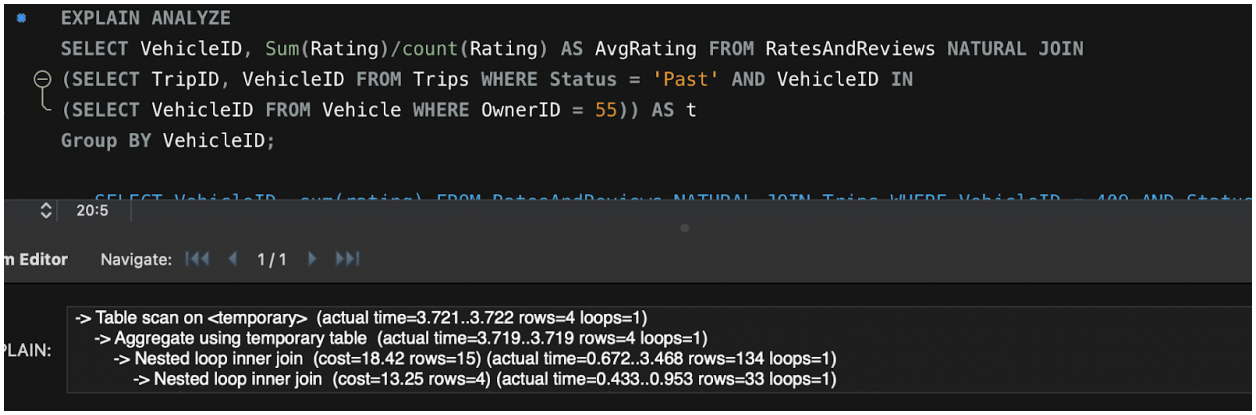
```
EXPLAIN ANALYZE
SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
(SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
(SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
Group BY VehicleID;
```

20:5

Filter Rows: Search Export:

EXPLAIN

-> Table scan on <temporary> (actual time=3.721..3.722 rows=4 loops=1) -> Aggregate using temporary ta…



```
EXPLAIN ANALYZE
SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
(SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
(SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
Group BY VehicleID;
```

20:5

Navigate: 1 / 1

```
-> Table scan on <temporary>  (actual time=3.721..3.722 rows=4 loops=1)
   -> Aggregate using temporary table  (actual time=3.719..3.719 rows=4 loops=1)
      -> Nested loop inner join  (cost=18.42 rows=15) (actual time=0.672..3.468 rows=134 loops=1)
         -> Nested loop inner join  (cost=13.25 rows=4) (actual time=0.433..0.953 rows=33 loops=1)
```

As we are not able to capture the entire results in screenshots, we have copied the results and pasted here:

As we can see,

*-> Table scan on <temporary> (actual time=**3.721..3.722** rows=4 loops=1)*

-> Aggregate using temporary table (actual time=3.719..3.719 rows=4 loops=1)

-> Nested loop inner join (cost=18.42 rows=15) (actual time=0.672..3.468 rows=134 loops=1)

-> Nested loop inner join (cost=13.25 rows=4) (actual time=0.433..0.953 rows=33 loops=1)

-> Covering index lookup on Vehicle using OweverID_idx (OwnerID=55) (cost=0.65 rows=4) (actual time=0.050..0.055 rows=4 loops=1)

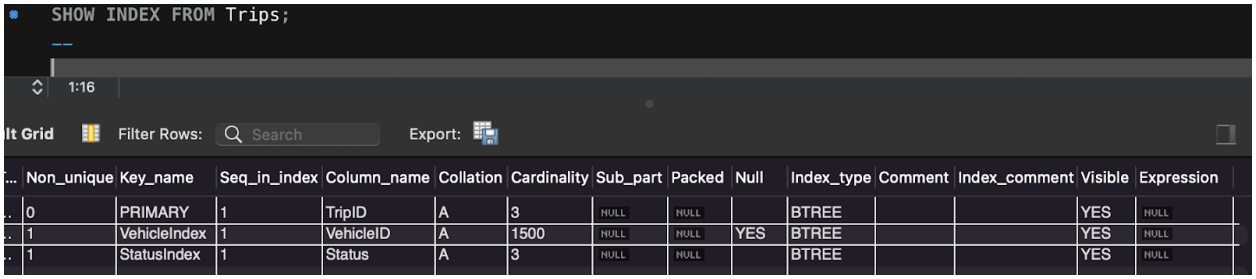
*-> Filter: (trips.`Status` = 'Past') (cost=2.27 rows=1) (actual time=**0.188..0.219** rows=8 loops=4)*

-> Index lookup on Trips using VehicleIndex (VehicleID=vehicle.VehicleID) (cost=2.27 rows=9) (actual time=0.183..0.208 rows=11 loops=4)

-> Index lookup on RatesAndReviews using TripID (TripID=trips.TripID) (cost=1.14 rows=4) (actual time=0.072..0.075 rows=4 loops=33)

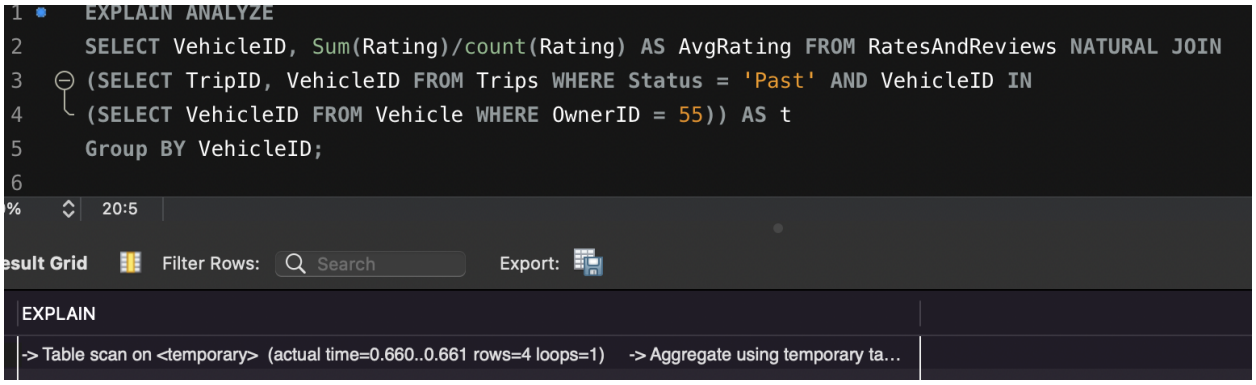
Here, for filtering out the past trips based on the status column, it took 0.031 seconds.

After Indexing on status column:



```
SHOW INDEX FROM Trips;
```

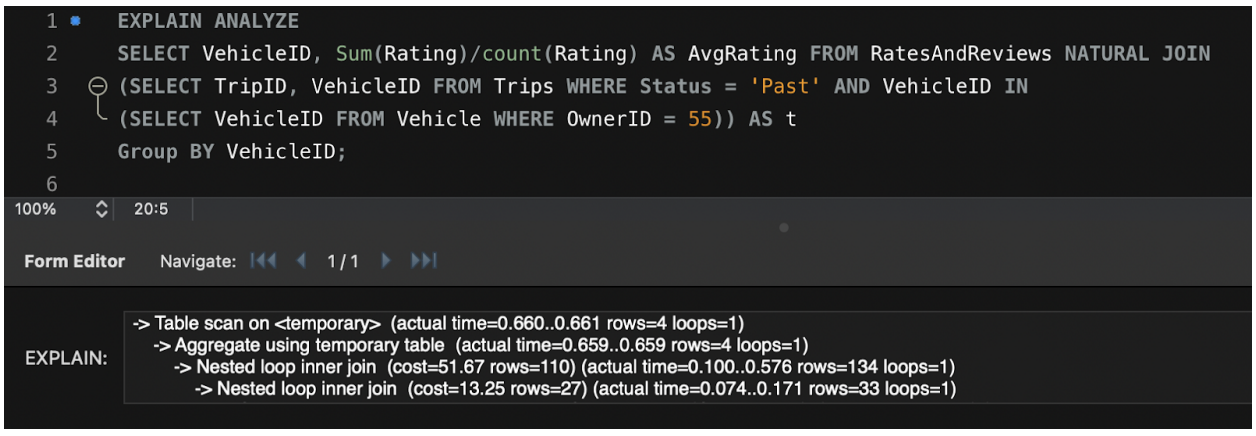
Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
0	PRIMARY	1	TripID	A	3	NULL	NULL		BTREE			YES	NULL
1	VehicleIndex	1	VehicleID	A	1500	NULL	NULL	YES	BTREE			YES	NULL
1	StatusIndex	1	Status	A	3	NULL	NULL		BTREE			YES	NULL



```
EXPLAIN ANALYZE
SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
(SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
(SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
Group BY VehicleID;
```

EXPLAIN

-> Table scan on <temporary> (actual time=0.660..0.661 rows=4 loops=1) -> Aggregate using temporary ta…



```
EXPLAIN ANALYZE
SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
(SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
(SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
Group BY VehicleID;
```

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Table scan on <temporary>  (actual time=0.660..0.661 rows=4 loops=1)
    -> Aggregate using temporary table  (actual time=0.659..0.659 rows=4 loops=1)
        -> Nested loop inner join  (cost=51.67 rows=110) (actual time=0.100..0.576 rows=134 loops=1)
            -> Nested loop inner join  (cost=13.25 rows=27) (actual time=0.074..0.171 rows=33 loops=1)
```

As we can see,

*-> Table scan on <temporary> (actual time=**0.660..0.661** rows=4 loops=1)*

-> Aggregate using temporary table (actual time=0.659..0.659 rows=4 loops=1)

-> Nested loop inner join (cost=51.67 rows=110) (actual time=0.100..0.576 rows=134 loops=1)

-> Nested loop inner join (cost=13.25 rows=27) (actual time=0.074..0.171 rows=33 loops=1)

-> Covering index lookup on Vehicle using OweverID_idx (OwnerID=55) (cost=0.65 rows=4) (actual time=0.024..0.025 rows=4 loops=1)

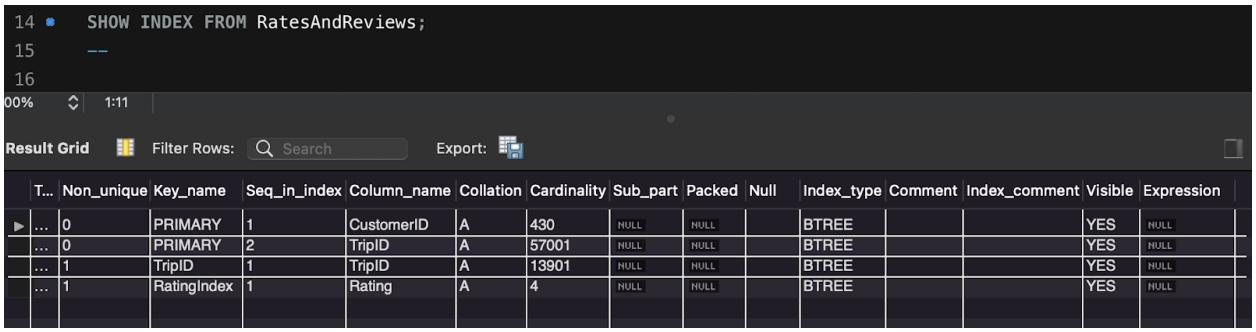
*-> Filter: (trips.`Status` = 'Past') (cost=2.42 rows=7) (actual time=**0.027..0.035** rows=8 loops=4)*

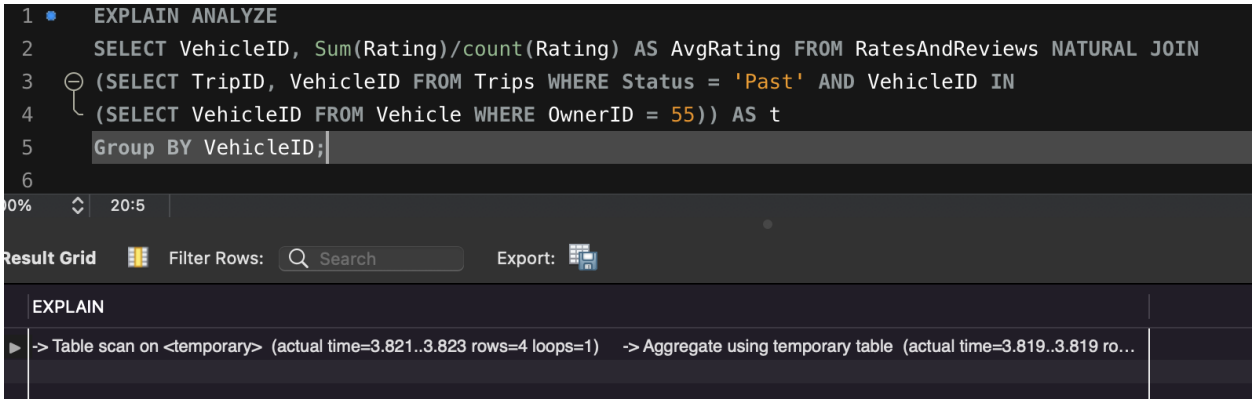
-> Index lookup on Trips using VehicleIndex (VehicleID=vehicle.VehicleID) (cost=2.42 rows=9) (actual time=0.027..0.033 rows=11 loops=4)

-> Index lookup on RatesAndReviews using TripID (TripID=trips.TripID) (cost=1.04 rows=4) (actual time=0.011..0.012 rows=4 loops=33)

Here, for filtering out the past trips based on the status column, it took 0.008 seconds.

After indexing on Rating column:





```
1  EXPLAIN ANALYZE
2  SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
3  (SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
4  (SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
5  Group BY VehicleID;
6
```

100% 20:5

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Table scan on <temporary>  (actual time=3.821..3.823 rows=4 loops=1)
    -> Aggregate using temporary table  (actual time=3.819..3.819 rows=4 loops=1)
        -> Nested loop inner join  (cost=18.42 rows=15) (actual time=0.699..3.562 rows=134 loops=1)
            -> Nested loop inner join  (cost=13.25 rows=4) (actual time=0.470..1.052 rows=33 loops=1)
```

As we can see,

-> Table scan on <temporary> (actual time=3.821..3.823 rows=4 loops=1)

-> Aggregate using temporary table (actual time=3.819..3.819 rows=4 loops=1)

-> Nested loop inner join (cost=18.42 rows=15) (actual time=0.699..3.562 rows=134 loops=1)

-> Nested loop inner join (cost=13.25 rows=4) (actual time=0.470..1.052 rows=33 loops=1)

-> Covering index lookup on Vehicle using OweverID_idx (OwnerID=55) (cost=0.65 rows=4) (actual time=0.067..0.071 rows=4 loops=1)

-> Filter: (trips.`Status` = 'Past') (cost=2.27 rows=1) (actual time=0.199..0.243 rows=8 loops=4)

-> Index lookup on Trips using VehicleIndex (VehicleID=vehicle.VehicleID) (cost=2.27 rows=9) (actual time=0.195..0.232 rows=11 loops=4)

-> Index lookup on RatesAndReviews using TripID (TripID=trips.TripID) (cost=1.14 rows=4) (actual time=0.072..0.075 rows=4 loops=33)

Here, there's no improvement in the performance as we're not focusing on any one particular value in the ratings field. Here, ratings field is an aggregation (sum) of chunks of ratings. So indexing on this field won't yield any significant results.

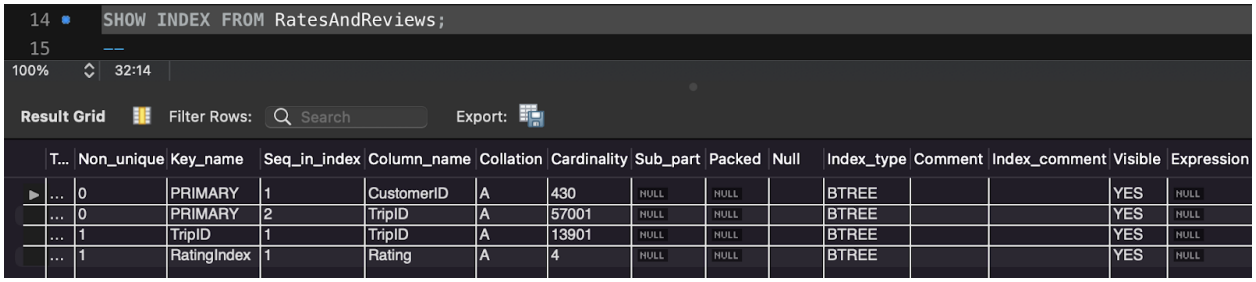
After indexing on both status and ratings fields:

```
14  SHOW INDEX FROM Trips;
15  --
```

100% 21:10

Result Grid Filter Rows: Search Export:

	T...	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
▶	...	0	PRIMARY	1	TripID	A	3	NULL	NULL		BTREE			YES	NULL
	...	1	VehicleIndex	1	VehicleID	A	1500	NULL	NULL	YES	BTREE			YES	NULL
	...	1	StatusIndex	1	Status	A	3	NULL	NULL		BTREE			YES	NULL

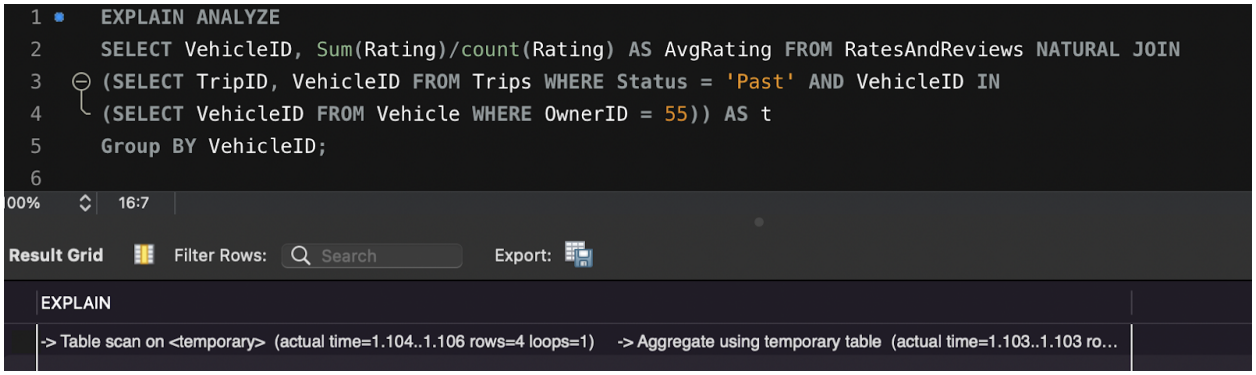


```
14  SHOW INDEX FROM RatesAndReviews;
15  --
```

100% 32:14

Result Grid Filter Rows: Search Export:

T...	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
...	0	PRIMARY	1	CustomerID	A	430	NULL	NULL		BTREE			YES	NULL
...	0	PRIMARY	2	TripID	A	57001	NULL	NULL		BTREE			YES	NULL
...	1	TripID	1	TripID	A	13901	NULL	NULL		BTREE			YES	NULL
...	1	RatingIndex	1	Rating	A	4	NULL	NULL		BTREE			YES	NULL

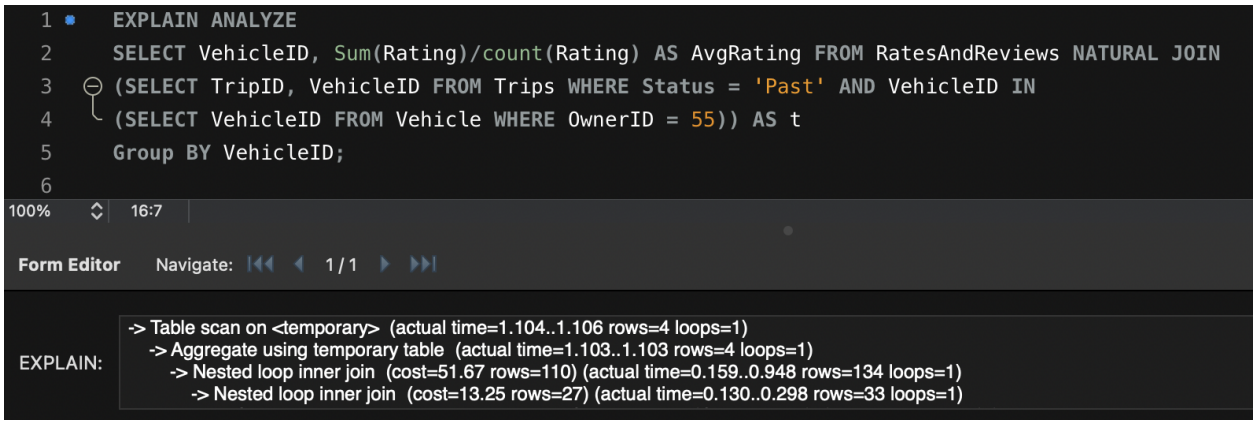


```
1  EXPLAIN ANALYZE
2  SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
3  (SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
4  (SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
5  Group BY VehicleID;
6
```

00% 16:7

Result Grid Filter Rows: Search Export:

EXPLAIN
-> Table scan on <temporary> (actual time=1.104..1.106 rows=4 loops=1) -> Aggregate using temporary table (actual time=1.103..1.103 ro...



```
1  EXPLAIN ANALYZE
2  SELECT VehicleID, Sum(Rating)/count(Rating) AS AvgRating FROM RatesAndReviews NATURAL JOIN
3  (SELECT TripID, VehicleID FROM Trips WHERE Status = 'Past' AND VehicleID IN
4  (SELECT VehicleID FROM Vehicle WHERE OwnerID = 55)) AS t
5  Group BY VehicleID;
6
```

100% 16:7

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Table scan on <temporary>  (actual time=1.104..1.106 rows=4 loops=1)
  -> Aggregate using temporary table  (actual time=1.103..1.103 rows=4 loops=1)
    -> Nested loop inner join  (cost=51.67 rows=110) (actual time=0.159..0.948 rows=134 loops=1)
      -> Nested loop inner join  (cost=13.25 rows=27) (actual time=0.130..0.298 rows=33 loops=1)
```

As we can see,

-> Table scan on <temporary> (actual time=1.104..1.106 rows=4 loops=1)

-> Aggregate using temporary table (actual time=1.103..1.103 rows=4 loops=1)

-> Nested loop inner join (cost=51.67 rows=110) (actual time=0.159..0.948 rows=134 loops=1)

-> Nested loop inner join (cost=13.25 rows=27) (actual time=0.130..0.298 rows=33 loops=1)

-> Covering index lookup on Vehicle using OweverID_idx (OwnerID=55) (cost=0.65 rows=4) (actual time=0.028..0.031 rows=4 loops=1)

-> Filter: (trips.`Status` = 'Past') (cost=2.42 rows=7) (actual time=0.049..0.065 rows=8 loops=4)

-> Index lookup on Trips using VehicleIndex (VehicleID=vehicle.VehicleID) (cost=2.42 rows=9) (actual time=0.048..0.059 rows=11 loops=4)

-> Index lookup on RatesAndReviews using TripID (TripID=trips.TripID) (cost=1.04 rows=4) (actual time=0.016..0.019 rows=4 loops=33)

Here, for filtering out the past trips based on the status column, it took 0.016 seconds.
This improvement is only because of status field and not ratings field, which proves indexing on ratings field is a bad idea.

SECOND COMPLEX QUERY

2) **When a user (seeker) gives a location and destination, an API will send the latitude and longitude of both. Based on these, we fetch all the trips that are happening in the same proximity and show them to the user.**
[Combination of Join of multiple relations and subqueries]

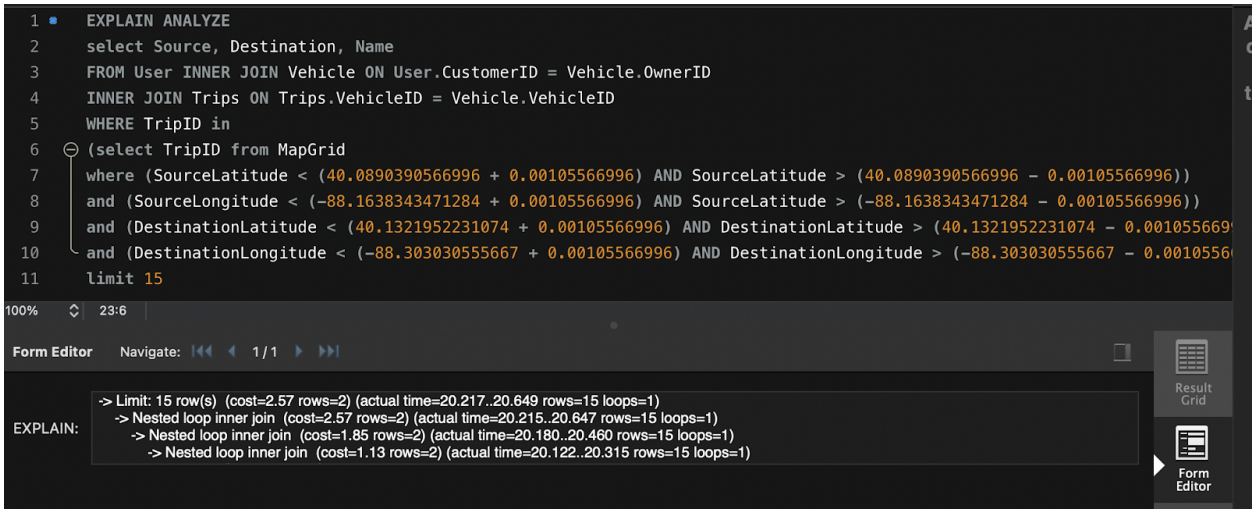
select Source, Destination, Name FROM User INNER JOIN Vehicle ON User.CustomerID = Vehicle.OwnerID INNER JOIN Trips ON Trips.VehicleID = Vehicle.VehicleID WHERE TripID in (select TripID from MapGrid where (SourceLatitude < (40.0890390566996 + 0.00105566996) AND SourceLatitude > (40.0890390566996 - 0.00105566996)) and (SourceLongitude < (-88.1638343471284 + 0.00105566996) AND SourceLatitude > (-88.1638343471284 - 0.00105566996)) and (DestinationLatitude < (40.1321952231074 + 0.00105566996) AND DestinationLatitude > (40.1321952231074 - 0.00105566996)) and (DestinationLongitude < (-88.303030555667 + 0.00105566996) AND DestinationLongitude > (-88.303030555667 - 0.00105566996))) limit 15;

Output screenshot:

Source	Destination	Name
2216 S Stonebrooke Ct Urbana	3404 Boulder Ridge Dr Champaign	Bobbie Gorczany
2216 S Stonebrooke Ct Urbana	3402 Boulder Ridge Dr Champaign	Mrs. Velma Cassin
2216 S Stonebrooke Ct Urbana	1518 GREYROCK LN Champaign	Deonte Schmeler MD
2216 S Stonebrooke Ct Urbana	1516 GREYROCK LN Champaign	Callie Turcotte
2214 S Stonebrooke Ct Urbana	3404 Boulder Ridge Dr Champaign	Lucie Corkery I
2214 S Stonebrooke Ct Urbana	3402 Boulder Ridge Dr Champaign	Braxton Konopelski
2214 S Stonebrooke Ct Urbana	1518 GREYROCK LN Champaign	Yesenia Willms I
2214 S Stonebrooke Ct Urbana	1516 GREYROCK LN Champaign	Ronaldo Gulgowski Sr.
3017 E Stone Creek Urbana, I...	3404 Boulder Ridge Dr Champaign	Hunter Heathcote
3017 E Stone Creek Urbana, I...	3402 Boulder Ridge Dr Champaign	Miss Teagan Marquardt DVM
3017 E Stone Creek Urbana, I...	1518 GREYROCK LN Champaign	Fannie Doyle
3017 E Stone Creek Urbana, I...	1516 GREYROCK LN Champaign	Alana Barton
2402 S St Andrews Urbana, IL...	3404 Boulder Ridge Dr Champaign	Mrs. Lizzie Mann
2402 S St Andrews Urbana, IL...	3402 Boulder Ridge Dr Champaign	Osvaldo Schowalter I
2402 S St Andrews Urbana, IL...	1518 GREYROCK LN Champaign	Dr. Reves Flatley Sr.

Before indexing(with Mysql's default indexing):

Before indexing:



-> Limit: 15 row(s) (cost=2.57 rows=2) (actual time=20.217..20.649 rows=15 loops=1)
-> Nested loop inner join (cost=2.57 rows=2) (actual time=20.215..20.647 rows=15 loops=1)
-> Nested loop inner join (cost=1.85 rows=2) (actual time=20.180..20.460 rows=15 loops=1)
-> Nested loop inner join (cost=1.13 rows=2) (actual time=20.122..20.315 rows=15 loops=1)
-> Filter: (`<subquery2>`.TripID is not null) (cost=695.23..0.41 rows=2) (actual time=20.067..20.071 rows=15 loops=1)
-> Table scan on <subquery2> (cost=1391.27..1392.58 rows=2) (actual time=20.066..20.068 rows=15 loops=1)
-> Materialize with deduplication (cost=1390.06..1390.06 rows=2) (actual time=20.064..20.064 rows=32 loops=1)
-> Filter: (mapgrid.TripID is not null) (cost=1389.85 rows=2) (actual time=0.108..20.026 rows=32 loops=1)
-> Filter: ((mapgrid.SourceLatitude < <cache>((40.0890390566996 + 0.00105566996))) and (mapgrid.SourceLatitude > <cache>((40.0890390566996 - 0.00105566996))) and (mapgrid.SourceLongitude < <cache>((-(88.1638343471284) + 0.00105566996))) and (mapgrid.SourceLatitude > <cache>((-(88.1638343471284) - 0.00105566996))) and (mapgrid.DestinationLatitude < <cache>((40.1321952231074 + 0.00105566996))) and (mapgrid.DestinationLatitude > <cache>((40.1321952231074 - 0.00105566996))) and (mapgrid.DestinationLongitude < <cache>((-(88.303030555667) + 0.00105566996))) and (mapgrid.DestinationLongitude > <cache>((-(88.303030555667) - 0.00105566996)))) (cost=1389.85 rows=2) (actual time=0.107..20.019 rows=32 loops=1)
-> Table scan on MapGrid (cost=1389.85 rows=13656) (actual time=0.086..17.848 rows=14115 loops=1)
-> Filter: (trips.VehicleID is not null) (cost=0.62 rows=1) (actual time=0.016..0.016 rows=1 loops=15)

-> Single-row index lookup on Trips using PRIMARY (TripID=`<subquery2>`.TripID) (cost=0.62 rows=1) (actual time=0.016..0.016 rows=1 loops=15)
-> Filter: (vehicle.OwnerID is not null) (cost=0.62 rows=1) (actual time=0.009..0.009 rows=1 loops=15)
-> Single-row index lookup on Vehicle using PRIMARY (VehicleID=trips.VehicleID) (cost=0.62 rows=1) (actual time=0.009..0.009 rows=1 loops=15)
-> Single-row index lookup on User using PRIMARY (CustomerID=vehicle.OwnerID) (cost=0.62 rows=1) (actual time=0.012..0.012 rows=1 loops=15)

After Indexing:

INDEXING ON SourceLAtitude:

CREATE INDEX SourceLatIndex
ON MapGrid(SourceLatitude);

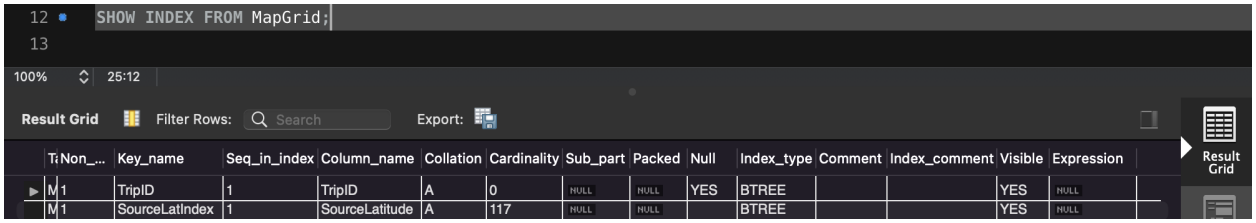
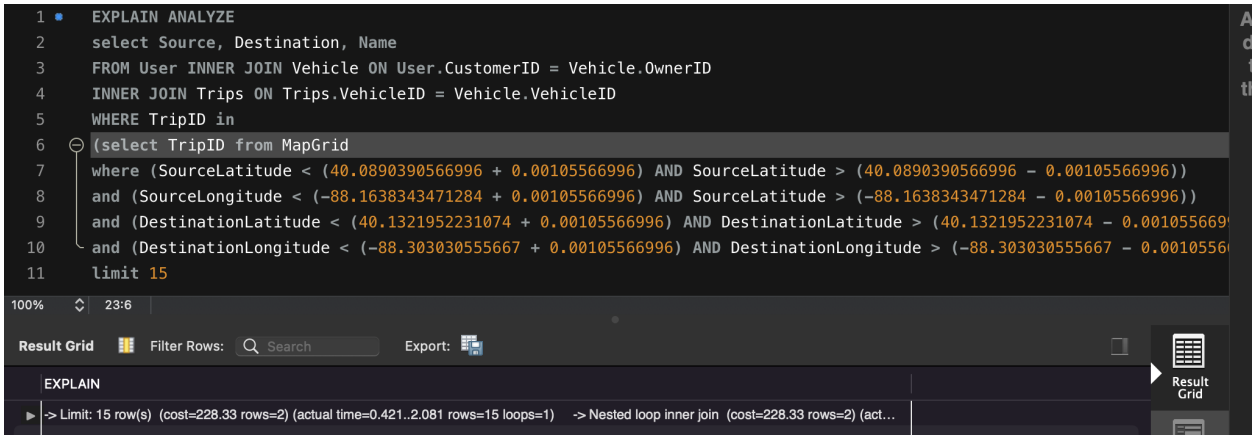
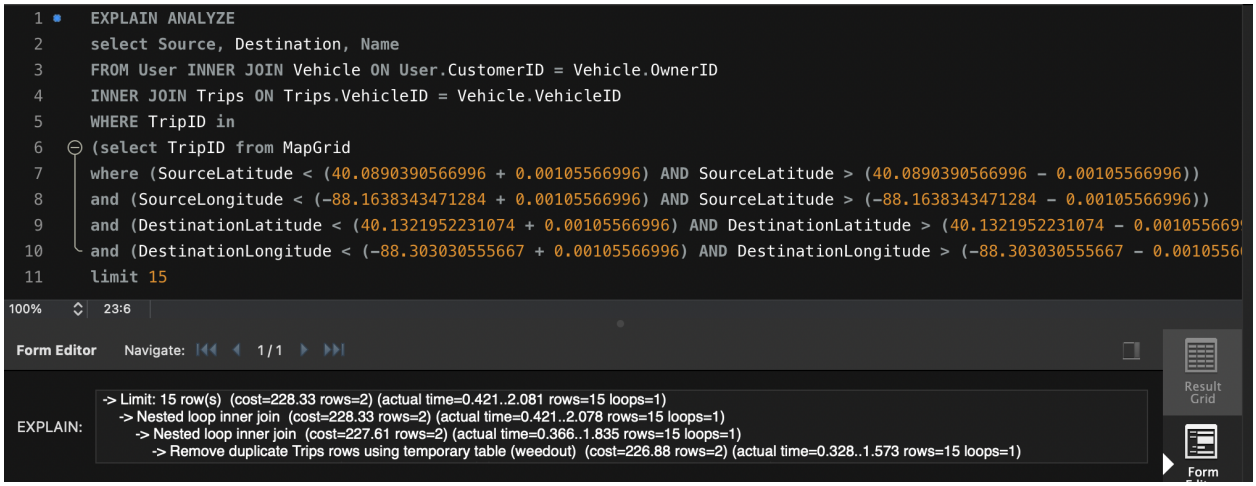


Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
M	1	TripID	1	TripID	A	0	NULL	NULL	YES	BTREE			YES	NULL
M	1	SourceLatIndex	1	SourceLatitude	A	117	NULL	NULL		BTREE			YES	NULL



EXPLAIN
-> Limit: 15 row(s) (cost=228.33 rows=2) (actual time=0.421..2.081 rows=15 loops=1) -> Nested loop inner join (cost=228.33 rows=2) (act…



-> Limit: 15 row(s) (cost=228.33 rows=2) (actual time=0.421..2.081 rows=15 loops=1)
    -> Nested loop inner join (cost=228.33 rows=2) (actual time=0.421..2.078 rows=15 loops=1)
    -> Nested loop inner join (cost=227.61 rows=2) (actual time=0.366..1.835 rows=15 loops=1)
    -> Remove duplicate Trips rows using temporary table (weedout) (cost=226.88 rows=2) (actual time=0.328..1.573 rows=15 loops=1)
        -> Nested loop inner join (cost=226.88 rows=2) (actual time=0.308..1.540 rows=15 loops=1)
        -> Filter: ((mapgrid.SourceLongitude < <cache>((-(88.1638343471284) + 0.00105566996))) and (mapgrid.DestinationLatitude < <cache>((40.1321952231074 + 0.00105566996))) and (mapgrid.DestinationLatitude > <cache>((40.1321952231074 - 0.00105566996))) and (mapgrid.DestinationLongitude < <cache>((-(88.303030555667) + 0.00105566996))) and (mapgrid.DestinationLongitude > <cache>((-(88.303030555667) - 0.00105566996))) and (mapgrid.TripID is not null)) (cost=226.16 rows=2) (actual time=0.253..1.204 rows=15 loops=1)
        -> Index range scan on MapGrid using SourceLatIndex over (40.0879833867396 < SourceLatitude < 40.0900947266596), with index condition: ((mapgrid.SourceLatitude < <cache>((40.0890390566996 + 0.00105566996))) and (mapgrid.SourceLatitude > <cache>((40.0890390566996 - 0.00105566996))) and (mapgrid.SourceLatitude > <cache>((-(88.1638343471284) - 0.00105566996)))) (cost=226.16 rows=502) (actual time=0.084..1.096 rows=255 loops=1)
        -> Filter: (trips.VehicleID is not null) (cost=0.30 rows=1) (actual time=0.022..0.022 rows=1 loops=15)
        -> Single-row index lookup on Trips using PRIMARY (TripID=mapgrid.TripID) (cost=0.30 rows=1) (actual time=0.021..0.021 rows=1 loops=15)

-> Filter: (vehicle.OwnerID is not null) (cost=0.30 rows=1) (actual time=0.017..0.017 rows=1 loops=15)
-> Single-row index lookup on Vehicle using PRIMARY (VehicleID=trips.VehicleID) (cost=0.30 rows=1) (actual time=0.016..0.017 rows=1 loops=15)
-> Single-row index lookup on User using PRIMARY (CustomerID=vehicle.OwnerID) (cost=0.30 rows=1) (actual time=0.016..0.016 rows=1 loops=15)

After indexing with sourceLatitude from the MapGrid table, there is a drastic execution time reduction. For each user we are trying to find all coordinates in it's proximity. Since, we use Source Latitude and Longitude for that, we try to index by SourceLatitude attribute.
This gave a significant improvement in time as there can be multiple entries for 1 latitude and indexing it has significantly improved the search time.Since we are using B trees and indexing them, it fetched faster.