Database connection:

```
mysql> aadya_doma@cloudshell:~ (cs411-pt1-378020)$ gcloud sql connect cs411pt1 --user=root --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 365
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

4 Main Database Tables

DDL Commands

CREATE TABLE FareTemp(fare_id VARCHAR(225), price VARCHAR(225), currency_type VARCHAR(225), payment_method VARCHAR(225), transfers VARCHAR(225), transfer_duration VARCHAR(225), PRIMARY KEY (fare_id));

CREATE TABLE ShapeTemp(shape_id VARCHAR(225), shape_pt_lat VARCHAR(225), shape_pt_lon VARCHAR(225), shape_pt_sequence VARCHAR(225), shape_dist_traveled VARCHAR(225), PRIMARY KEY (shape_id, shape_pt_sequence));

CREATE TABLE TripTemp(route_id VARCHAR(225),service_id VARCHAR(225),trip_id VARCHAR(225), trip_headsign VARCHAR(225), direction_id VARCHAR(225), shape_id VARCHAR(225), PRIMARY KEY (trip_id));

CREATE TABLE Route1(route_id VARCHAR(225), agency_id VARCHAR(225), route_short_name VARCHAR(225), route_long_name VARCHAR(225), route_type VARCHAR(225), route_color VARCHAR(225), route_text_color VARCHAR(225), PRIMARY KEY (route_id));

CREATE TABLE StopTimeTemp(trip_id VARCHAR(225), arrival_time VARCHAR(225), departure_time VARCHAR(225), stop_id VARCHAR(225), stop_sequence VARCHAR(225), PRIMARY KEY (trip_id, arrival_time, departure_time));

CREATE TABLE StopTemp(stop_id INT, stop_name VARCHAR(225), stop_desc VARCHAR(225), stop_lat REAL, stop_lon REAL);

CREATE TABLE AmountToPay(FareID INT NOT NULL, TripID INT NOT NULL, ArrivalTime hh:mm:ss, DepartureTime hh:mm:ss, PRIMARY KEY (FareID, TripID));

CREATE TABLE contains (StopID INT NOT NULL, RouteID INT NOT NULL, PRIMARY KEY (StopID, RouteID));

As shown below, there are at least 1000 rows each in the following tables:

```
mysql> select count(*) from ShapeTemp;
+----------+
| count(*) |
+----------+
|   564392 |
+----------+
1 row in set (0.03 sec)
```

```
mysql> select count(*) from Route1;
+----------+
| count(*) |
+----------+
|      680 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from TripTemp;
+----------+
| count(*) |
+----------+
|     1114 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from StopTemp;
+----------+
| count(*) |
+----------+
|    20902 |
+----------+
1 row in set (0.01 sec)

mysql>
```

```
mysql> select count(*) from StopTimeTemp;
+----------+
| count(*) |
+----------+
|    47633 |
+----------+
1 row in set (0.01 sec)
```

2 Advanced SQL Queries

1. Output all trips that stop at a stop that contains the name Santana or have agency id 1. Return type: trip_id

```
SELECT trip_id
FROM StopTemp NATURAL JOIN StopTimeTemp NATURAL JOIN TripTemp
WHERE stop_name = '%Santana%'
UNION
SELECT trip_id
FROM Route1 NATURAL JOIN TripTemp
WHERE agency_id = 1;
```

```
| trip_id      |
+--------------+
| 1012-10-0    |
| 1016-10-1    |
| 1018-10-1    |
| 1020-10-1    |
| 1025-10-0    |
| 1034-10-0    |
| 106A-10-0    |
| 1156-10-0    |
| 1177-31-0    |
| 1178-31-0    |
| 118C-10-1    |
| 119C-10-1    |
| 119L-10-1    |
| 121G-10-0    |
| 129F-10-1    |
| 139A-10-1    |
| 148P-10-1    |
| 1701-21-0    |
| 1702-10-0    |
```

2. Count the number of trips at each stop whose stop name starts with Av. (Return type: stop_id and the corresponding counts of each trip_id). Only output the stops that have a count larger than 4

```
SELECT stop_id, COUNT(trip_id) as count
FROM StopTemp NATURAL JOIN StopTimeTemp NATURAL JOIN TripTemp
WHERE stop_name LIKE 'Av.%'
GROUP BY stop_id
HAVING count >= 4;
```

stop_id	count
840000676	8
830004435	6
410003574	7
410003577	7
410003582	6
710000978	6
710000597	4
840000586	4
840003991	4
830003996	4
640000224	5
640000372	4
640000368	4
6415247	4
640000370	4
640000366	4
6414087	4
7113134	6
90015086	5
90008382	5
710000969	5
710017213	4
8812651	4
940003767	8
9412668	8
940004420	5
940003781	4
910003785	4
910000893	4
910000862	5

Indexing

First Query:

EXPLAIN ANALYZE before Index:

```
| -> Table scan on <union temporary>  (cost=0.01..34.69 rows=2576) (actual time=0.002..0.031 rows=527 loops=1)
    -> Union materialize with deduplication  (cost=12249.48..12284.15 rows=2576) (actual time=55.414..55.472 rows=527 loops=1)
        -> Nested loop inner join  (cost=11488.82 rows=2465) (actual time=51.715..51.715 rows=0 loops=1)
            -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.068..18.267 rows=23884 loops=1)
                -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.034..0.434 rows=1114 loops=1)
                -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.008..0.015 rows=21 loops=1114)
                    -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.008..0.013 rows=21 loops=1114)
            -> Filter: ((StopTemp.stop_name = '%Santana%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001..0
.001 rows=0 loops=23884)
                -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
        -> Nested loop inner join  (cost=503.05 rows=111) (actual time=0.069..3.441 rows=527 loops=1)
            -> Filter: (TripTemp.route_id is not null)  (cost=113.15 rows=1114) (actual time=0.054..0.513 rows=1114 loops=1)
                -> Table scan on TripTemp  (cost=113.15 rows=1114) (actual time=0.053..0.414 rows=1114 loops=1)
            -> Filter: (Route1.agency_id = 1)  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=1114)
                -> Single-row index lookup on Route1 using PRIMARY (route_id=TripTemp.route_id)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=0 loops=1114)
|
```

Index:

- CREATE INDEX idxOne ON StopTemp(stop_id);
- CREATE INDEX idxTwo ON Route1(agency_id);
- CREATE INDEX idxThree ON StopTemp(stop_name);

EXPLAIN ANALYZE after idxOne:

```
| -> Table scan on <union temporary>  (cost=0.01..34.69 rows=2576) (actual time=0.001..0.031 rows=527 loops=1)
    -> Union materialize with deduplication  (cost=12249.48..12284.15 rows=2576) (actual time=56.561..56.625 rows=527 loops=1)
        -> Nested loop inner join  (cost=11488.82 rows=2465) (actual time=53.226..53.226 rows=0 loops=1)
            -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.063..19.424 rows=23884 loops=1)
                -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.038..0.459 rows=1114 loops=1)
                -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.009..0.016 rows=21 loops=1114)
                    -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.009..0.014 rows=21 loops=1114)
            -> Filter: ((StopTemp.stop_name = '%Santana%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001..0
.001 rows=0 loops=23884)
                -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
        -> Nested loop inner join  (cost=503.05 rows=111) (actual time=0.048..3.122 rows=527 loops=1)
            -> Filter: (TripTemp.route_id is not null)  (cost=113.15 rows=1114) (actual time=0.033..0.454 rows=1114 loops=1)
                -> Table scan on TripTemp  (cost=113.15 rows=1114) (actual time=0.032..0.365 rows=1114 loops=1)
            -> Filter: (Route1.agency_id = 1)  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=1114)
                -> Single-row index lookup on Route1 using PRIMARY (route_id=TripTemp.route_id)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=0 loops=1114)
|
```

EXPLAIN ANALYZE after idxTwo:

```
| -> Table scan on <union temporary>  (cost=0.01..34.69 rows=2576) (actual time=0.001..0.031 rows=527 loops=1)
    -> Union materialize with deduplication  (cost=12249.48..12284.15 rows=2576) (actual time=52.672..52.732 rows=527 loops=1)
        -> Nested loop inner join  (cost=11488.82 rows=2465) (actual time=49.374..49.374 rows=0 loops=1)
            -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.062..17.085 rows=23884 loops=1)
                -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.034..0.403 rows=1114 loops=1)
                -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.007..0.014 rows=21 loops=1114)
                    -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.007..0.012 rows=21 loops=1114)
            -> Filter: ((StopTemp.stop_name = '%Santana%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001..0
.001 rows=0 loops=23884)
                -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
        -> Nested loop inner join  (cost=503.05 rows=111) (actual time=0.049..3.086 rows=527 loops=1)
            -> Filter: (TripTemp.route_id is not null)  (cost=113.15 rows=1114) (actual time=0.028..0.434 rows=1114 loops=1)
                -> Table scan on TripTemp  (cost=113.15 rows=1114) (actual time=0.026..0.345 rows=1114 loops=1)
            -> Filter: (Route1.agency_id = 1)  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=1114)
                -> Single-row index lookup on Route1 using PRIMARY (route_id=TripTemp.route_id)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=0 loops=1114)
|
```

EXPLAIN ANALYZE after idxThree:

```
| -> Table scan on <union temporary>  (cost=0.01..34.69 rows=2576) (actual time=0.001..0.032 rows=527 loops=1)
    -> Union materialize with deduplication  (cost=3624.77..3659.44 rows=2576) (actual time=3.592..3.652 rows=527 loops=1)
        -> Nested loop inner join  (cost=2864.12 rows=2465) (actual time=0.023..0.023 rows=0 loops=1)
            -> Inner hash join (no condition)  (cost=114.25 rows=1114) (actual time=0.022..0.022 rows=0 loops=1)
                -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (never executed)
                -> Hash
                    -> Index lookup on StopTemp using idxThree (stop_name='%Santana%')  (cost=1.10 rows=1) (actual time=0.015..0.015 rows=0 loops=1)
            -> Filter: (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double))  (cost=0.26 rows=2) (never executed)
                -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (never executed)
        -> Nested loop inner join  (cost=503.05 rows=111) (actual time=0.045..3.341 rows=527 loops=1)
            -> Filter: (TripTemp.route_id is not null)  (cost=113.15 rows=1114) (actual time=0.031..0.461 rows=1114 loops=1)
                -> Table scan on TripTemp  (cost=113.15 rows=1114) (actual time=0.030..0.368 rows=1114 loops=1)
            -> Filter: (Route1.agency_id = 1)  (cost=0.25 rows=0) (actual time=0.002..0.002 rows=0 loops=1114)
                -> Single-row index lookup on Route1 using PRIMARY (route_id=TripTemp.route_id)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=0 loops=1114)
|
```

Analysis:

For idx_1, we added an index on stop_id of StopTemp. The reason we chose idx_1 is we thought it might have improved our cost because our natural join for StopTemp and StopTimeTemp is on stop_id, which we thought could have potentially increased the speed at which the query could

iterate over our tables. As shown from EXPLAIN ANALYZE above, this index did not improve the query time significantly- In fact, it increased the search time from 55.414 to 56.561. Thus, we did not choose this as our final index. For idx_2, we added an index on agency_id of Route1. The reason we chose idx_2 is because of the reference in the where clause to agency_id in the second part of the union. We suspected that if we indexed by agency_id, it would make searching for data where agency_id = 1 significantly more efficient; thereby eliminating the need for the WHERE clause. As shown from EXPLAIN ANALYZE above, this index did not improve the query time significantly and only minimally decreased the filter time from 55.414 to 52.672. In contrast, for idx_3, we added an index on stop_name of StopTemp. We chose this index because of the reference to stop_name in the where clause of the first part of the union. This index significantly improved the search time of the query, decreasing it from 55.414 to 3.592. Thus, we will choose this index as our final index for the first query, as it greatly improves the query performance.

<u>Second Query:</u>

EXPLAIN ANALYZE before Index:

```
-------------------------+
| -> Filter: (count >= 4)  (actual time=212.257..213.001 rows=832 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.196 rows=4327 loops=1)
        -> Aggregate using temporary table  (actual time=212.253..212.698 rows=4327 loops=1)
            -> Nested loop inner join  (cost=20016.92 rows=2738) (actual time=1.981..205.806 rows=9998 loops=1)
                -> Nested loop inner join  (cost=3360.18 rows=24645) (actual time=1.923..114.736 rows=23884 loops=1)
                    -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.041..0.679 rows=1114 loops=1)
                    -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.70 rows=22) (actual time=0.089..0.101 rows=21 loops=1114)
                        -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.70 rows=22) (actual time=0.089..0.098 rows=21 loops=1114)
                -> Filter: ((StopTemp.stop_name like 'Av.%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.58 rows=0) (actual time=0.004.
.0.004 rows=0 loops=23884)
                    -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.58 rows=1) (actual time=0.003..0.003 rows=1 loops=23884)
 |
```

Index:

- CREATE INDEX idxFour ON StopTimeTemp(trip_id);
- CREATE INDEX idxFive ON StopTemp(stop_id);
- CREATE INDEX idxSix ON StopTemp(stop_name);

EXPLAIN ANALYZE after idxFour:

```
| -> Filter: (count >= 4)  (actual time=58.251..59.008 rows=832 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.196 rows=4327 loops=1)
        -> Aggregate using temporary table  (actual time=58.247..58.691 rows=4327 loops=1)
            -> Nested loop inner join  (cost=11488.82 rows=2738) (actual time=0.086..53.925 rows=9998 loops=1)
                -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.057..17.829 rows=23884 loops=1)
                    -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.035..0.401 rows=1114 loops=1)
                    -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.007..0.014 rows=21 loops=1114)
                        -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.007..0.012 rows=21 loops=1114)
                -> Filter: ((StopTemp.stop_name like 'Av.%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001.
.0.001 rows=0 loops=23884)
                    -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
 |
```

EXPLAIN ANALYZE after idxFive:

```
| -> Filter: (count >= 4)  (actual time=59.011..59.763 rows=832 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.201 rows=4327 loops=1)
        -> Aggregate using temporary table  (actual time=59.007..59.457 rows=4327 loops=1)
            -> Nested loop inner join  (cost=11488.82 rows=2738) (actual time=0.096..54.596 rows=9998 loops=1)
                -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.062..18.049 rows=23884 loops=1)
                    -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.036..0.427 rows=1114 loops=1)
                    -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.007..0.014 rows=21 loops=1114)
                        -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.007..0.012 rows=21 loops=1114)
                -> Filter: ((StopTemp.stop_name like 'Av.%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001.
.0.001 rows=0 loops=23884)
                    -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
 |
```

EXPLAIN ANALYZE after idxSix:

```
| -> Filter: (count >= 4)  (actual time=58.333..59.062 rows=832 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.187 rows=4327 loops=1)
        -> Aggregate using temporary table  (actual time=58.329..58.763 rows=4327 loops=1)
            -> Nested loop inner join  (cost=11488.82 rows=12322) (actual time=0.090..53.960 rows=9998 loops=1)
                -> Nested loop inner join  (cost=2863.01 rows=24645) (actual time=0.059..17.463 rows=23884 loops=1)
                    -> Index scan on TripTemp using PRIMARY  (cost=113.15 rows=1114) (actual time=0.035..0.456 rows=1114 loops=1)
                    -> Filter: (StopTimeTemp.stop_id is not null)  (cost=0.26 rows=22) (actual time=0.007..0.014 rows=21 loops=1114)
                        -> Index lookup on StopTimeTemp using PRIMARY (trip_id=TripTemp.trip_id)  (cost=0.26 rows=22) (actual time=0.007..0.012 rows=21 loops=1114)
                -> Filter: ((StopTemp.stop_name like 'Av.%') and (cast(StopTemp.stop_id as double) = cast(StopTimeTemp.stop_id as double)))  (cost=0.25 rows=0) (actual time=0.001.
.0.001 rows=0 loops=23884)
                    -> Single-row index lookup on StopTemp using PRIMARY (stop_id=StopTimeTemp.stop_id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=23884)
 |
```

Analysis:

For idxFour, we added an index on trip_id of StopTimeTemp. As shown from EXPLAIN ANALYZE above, this index did improve the query time significantly and decreased the search time from 212.257 to 58.251. For idxFive, we added an index on stop_id of StopTemp.We chose this attribute as we believed that if we indexed by this attribute in our GROUP BY, it would be easier for the query to search for our data (as the tables would be more efficiently stored). This index improved the query time but not as much as idxFour, decreasing the filter time from 212.257 to 59.763. Lastly, for idxSix, we added an index on stop_name of StopTemp. This index improved the search time of the query as well, decreasing it from 212.257 to 58.333 (about the same as idxFour). Thus, we will likely choose idxFour and idxSix as our final indexes for the second query. We believe that index 4 and index 6 were better performing as it made it easier for the query to check if a single select met the conditions to be successful, while index 5 did not do this and simply made counting more efficient for the count clause (which the query would have to count for anyway).