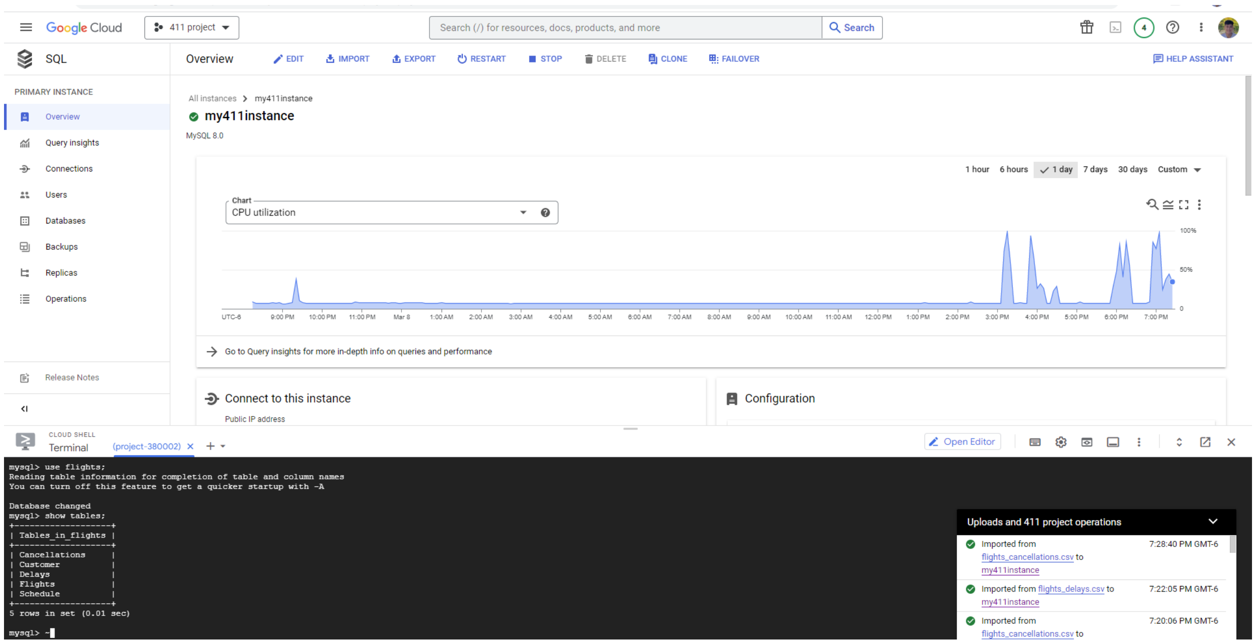
CHANGES SINCE STAGE 3 DUE DATE:

- fixed advanced queries to include aggregation and join feature

NOTE:

- in stage 3 grading feedback, the TA asked if our FlightID's were unique. Our FlightID's ARE unique because we generated our own FlightIDs when loading in the data by giving FlightID the auto_increment attribute (instead of using the FlightIDs from the dataset).

Database implementation:



```
mysql> use flights;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------+
| Tables_in_flights |
+-------------------+
| Cancellations     |
| Customer          |
| Delays            |
| Flights           |
| Schedule          |
+-------------------+
5 rows in set (0.01 sec)

mysql> ~
```

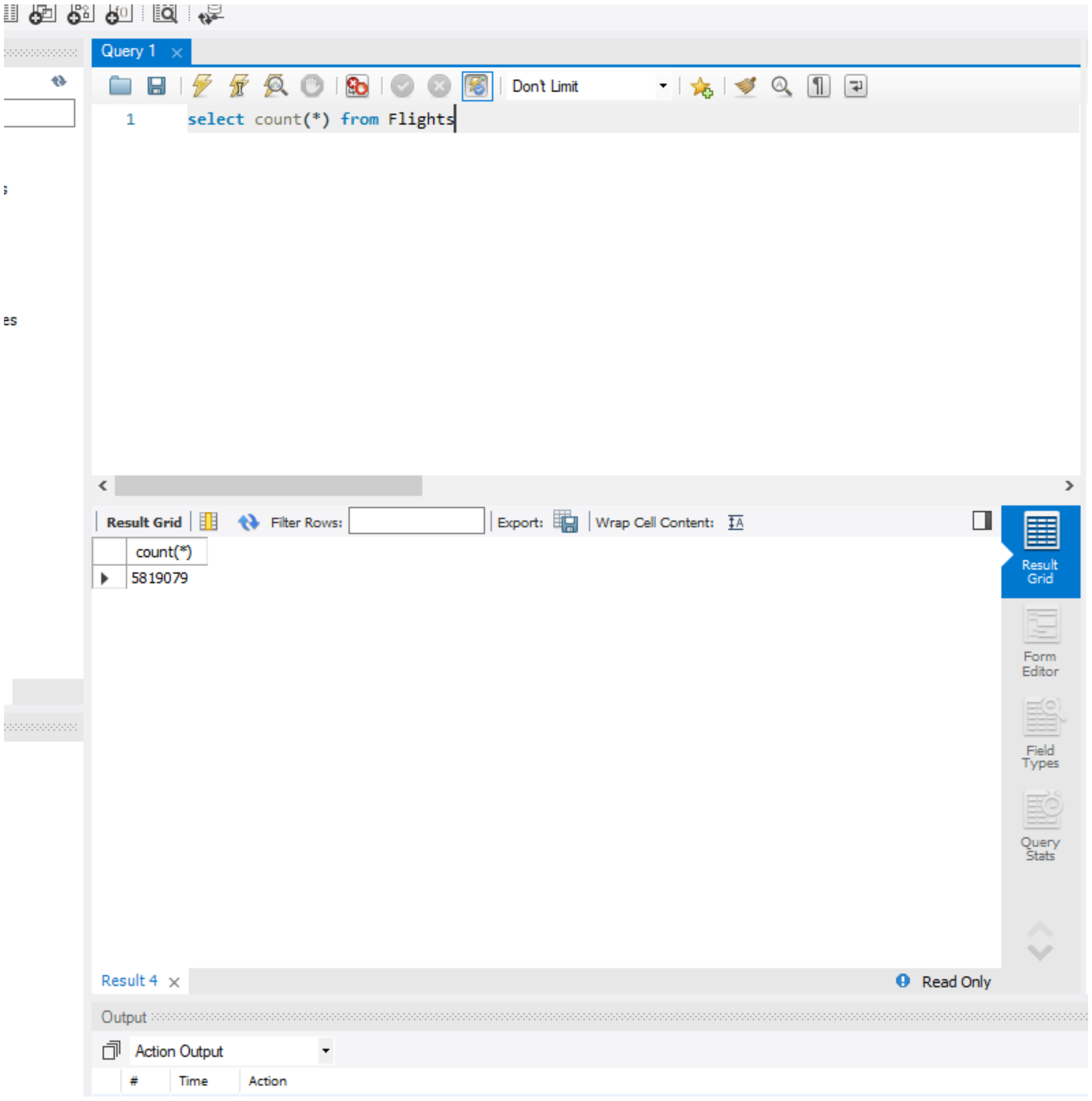
Uploads and 411 project operations

Imported from flights_cancellations.csv to my411instance	7:28:40 PM GMT-6
Imported from flights_delays.csv to my411instance	7:22:05 PM GMT-6
Imported from flights_cancellations.csv to	7:20:06 PM GMT-6

DDL commands:

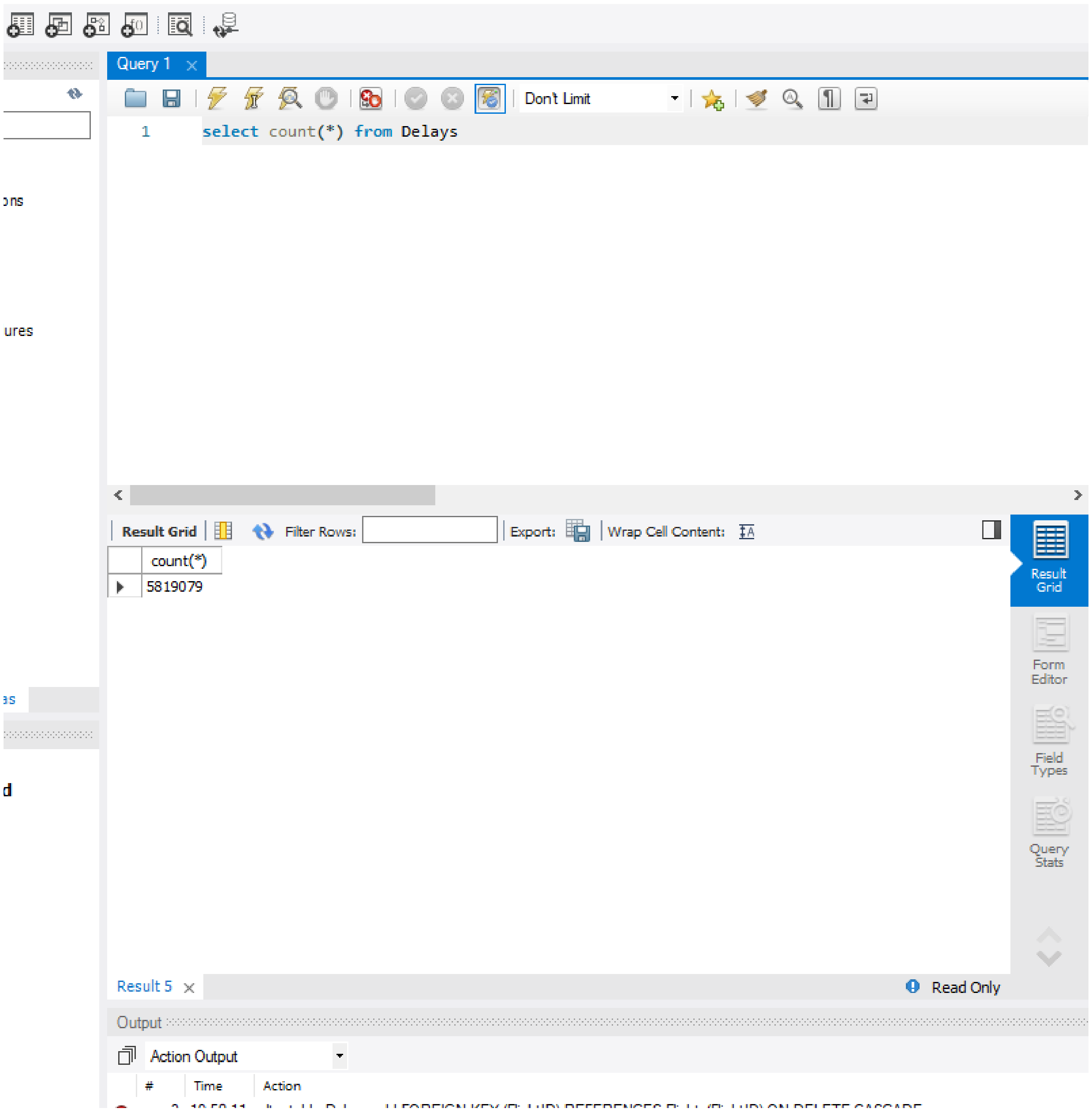
```
CREATE TABLE Flights (
   FlightId INT AUTO_INCREMENT,
   Airline VARCHAR(50),
   Origin VARCHAR(5),
   Destination VARCHAR(5),
   Date INT,
   Month INT,
   PRIMARY KEY(FlightId)
);

CREATE TABLE Customer (
   CustomerId INT,
   Username VARCHAR(50),
   Password_ VARCHAR(25),
   PRIMARY KEY(CustomerId)
);
```

```
CREATE TABLE Schedule (
   FlightId INT AUTO_INCREMENT,
   CustomerId INT,
   PRIMARY KEY(FlightId, CustomerId),
   FOREIGN KEY (FlightID) REFERENCES Flights(FlightID) ON DELETE CASCADE,
   FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId) ON DELETE
CASCADE
);

CREATE TABLE Cancellations (
   FlightId INT AUTO_INCREMENT,
   Diverted INT,
   Cancelled INT,
   PRIMARY KEY(FlightId),
   FOREIGN KEY (FlightID) REFERENCES Flights(FlightID)
      ON DELETE CASCADE
);

CREATE TABLE Delays (
   FlightId INT AUTO_INCREMENT,
   DepartureDelay INT,
   ArrivalDelay INT,
   PRIMARY KEY(FlightId),
   FOREIGN KEY (FlightID) REFERENCES Flights(FlightID)
      ON DELETE CASCADE
);
```



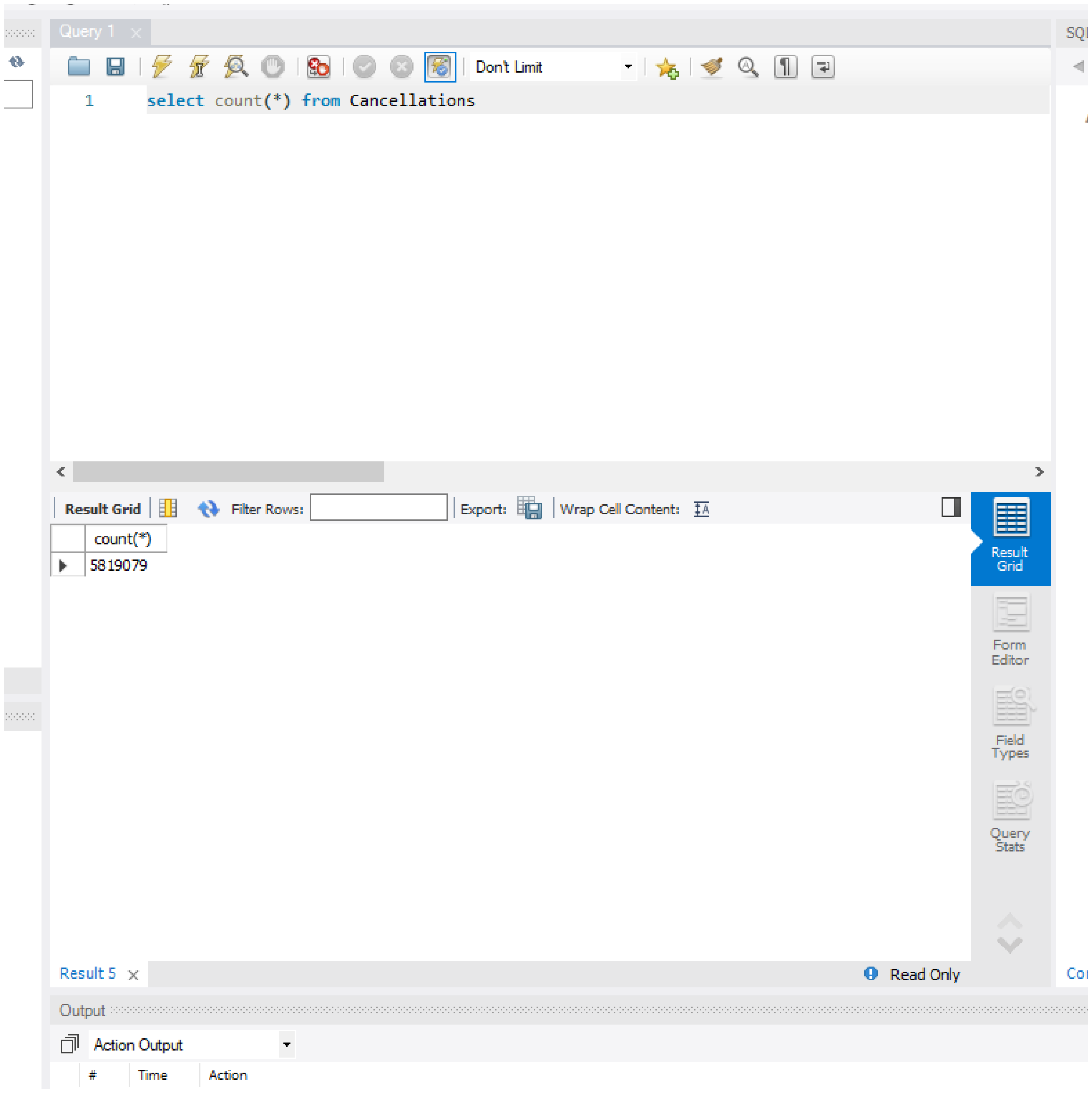
```
select count(*) from Flights
```

```
count(*)
5819079
```



```sql
select count(*) from Delays
```

count(*)
5819079

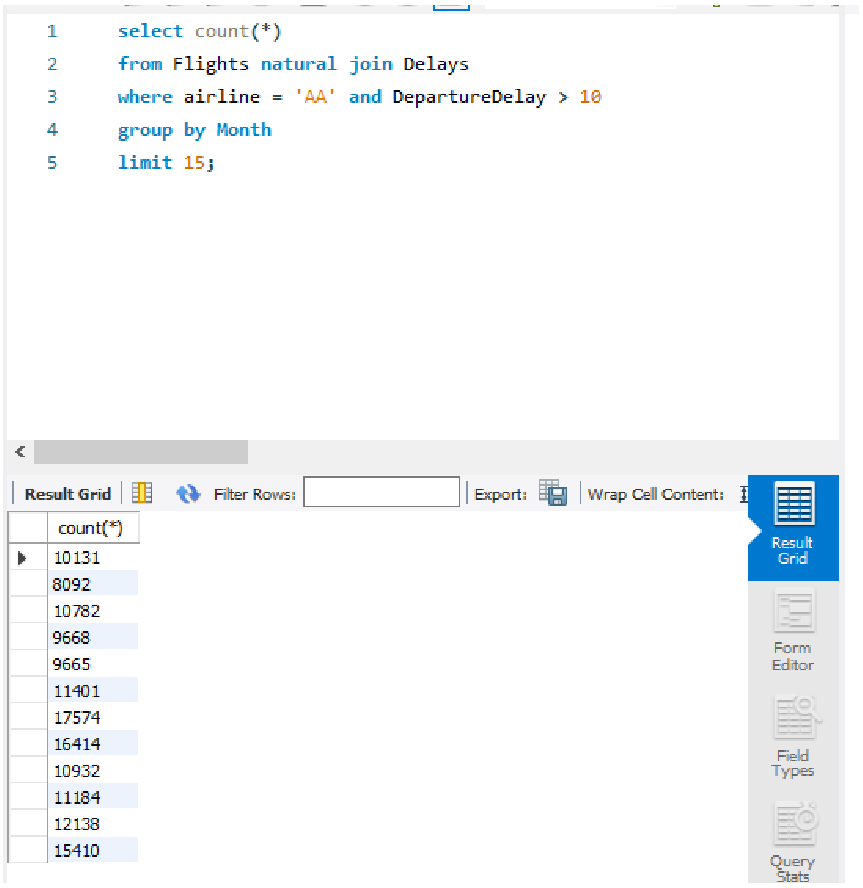


```
select count(*) from Cancellations
```

```
count(*)
5819079
```

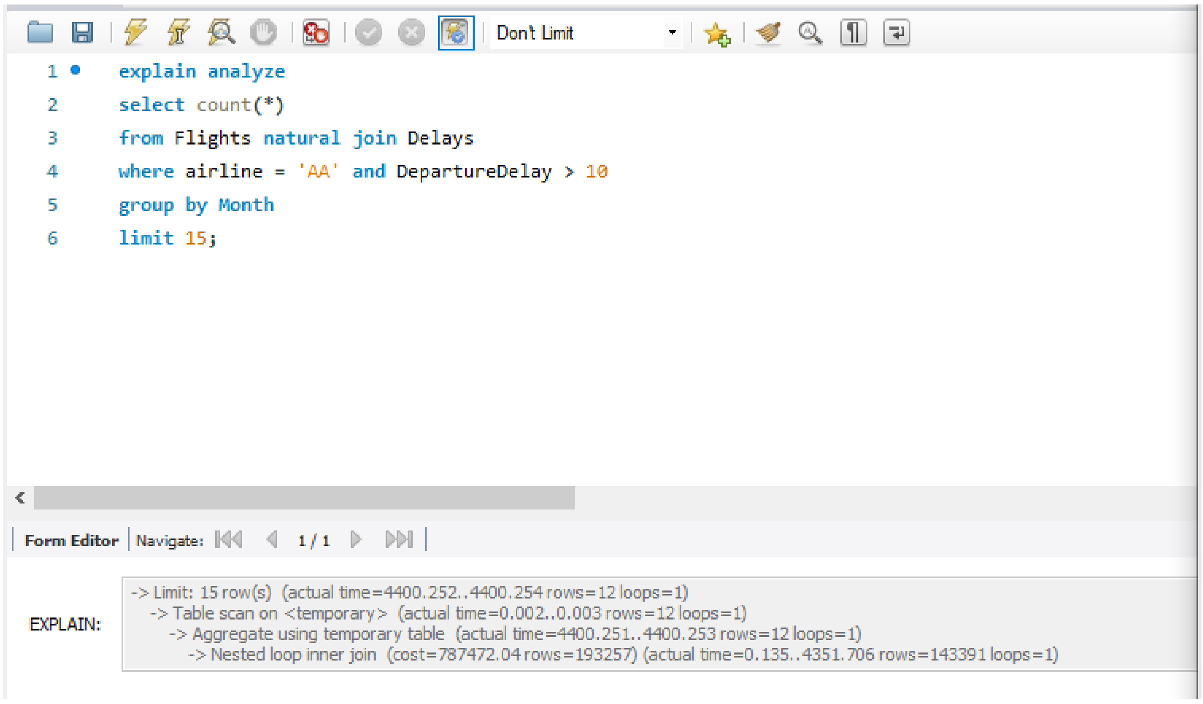
Indexing:

query 1:



```
select count(*)
from Flights natural join Delays
where airline = 'AA' and DepartureDelay > 10
group by Month
limit 15;
```

count(*)
10131
8092
10782
9668
9665
11401
17574
16414
10932
11184
12138
15410

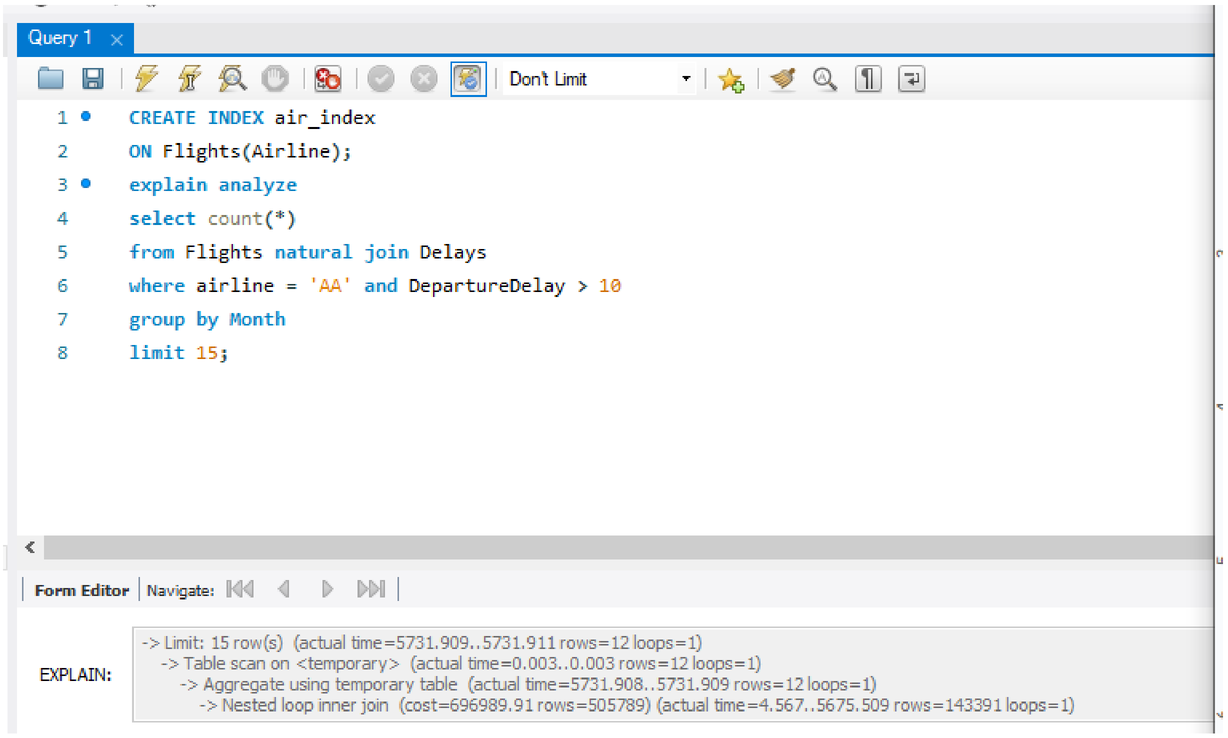


```
explain analyze
select count(*)
from Flights natural join Delays
where airline = 'AA' and DepartureDelay > 10
group by Month
limit 15;
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=4400.252..4400.254 rows=12 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.003 rows=12 loops=1)
        -> Aggregate using temporary table  (actual time=4400.251..4400.253 rows=12 loops=1)
            -> Nested loop inner join  (cost=787472.04 rows=193257) (actual time=0.135..4351.706 rows=143391 loops=1)
```

After indexing on Airline name: cost decreased from around 800 to 700k. Chose this index because the query is filtering by airline, so indexing would make it faster for query to filter.

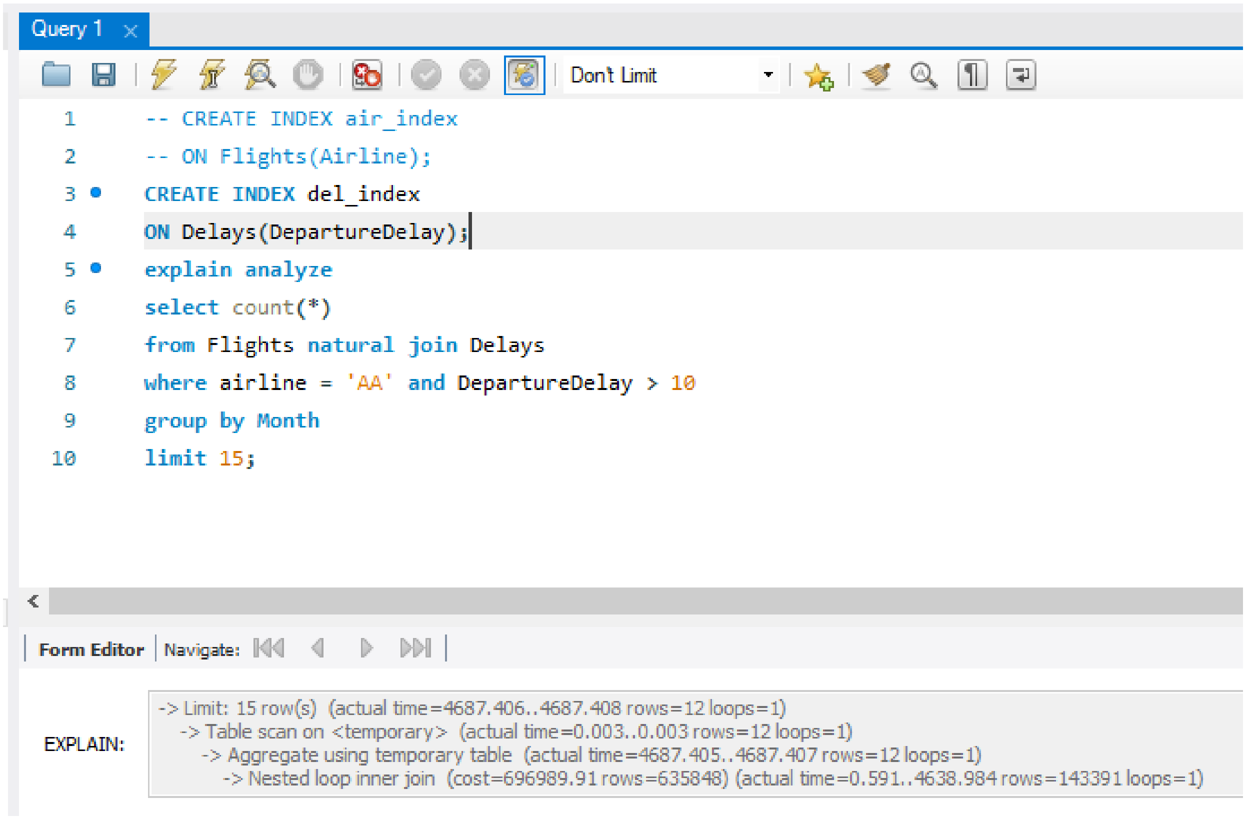


```sql
CREATE INDEX air_index
ON Flights(Airline);
explain analyze
select count(*)
from Flights natural join Delays
where airline = 'AA' and DepartureDelay > 10
group by Month
limit 15;
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=5731.909..5731.911 rows=12 loops=1)
    -> Table scan on <temporary>  (actual time=0.003..0.003 rows=12 loops=1)
        -> Aggregate using temporary table  (actual time=5731.908..5731.909 rows=12 loops=1)
            -> Nested loop inner join  (cost=696989.91 rows=505789) (actual time=4.567..5675.509 rows=143391 loops=1)
```

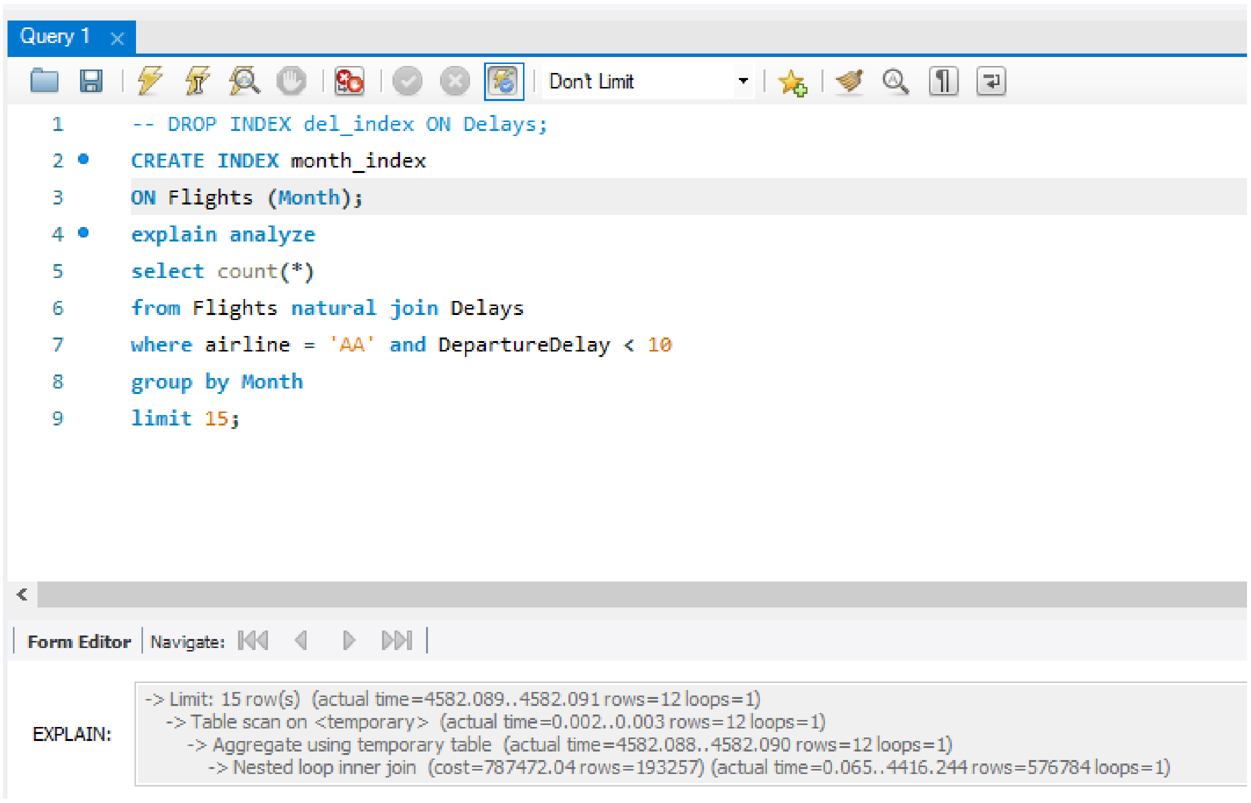
After indexing on Airline name and Delay time: cost didn’t change when compared to only indexing on Airline name. Chose this index since our query also filters on delay time, so we thought indexing on delay time would decrease the cost



```
-- CREATE INDEX air_index
-- ON Flights(Airline);
CREATE INDEX del_index
ON Delays(DepartureDelay);
explain analyze
select count(*)
from Flights natural join Delays
where airline = 'AA' and DepartureDelay > 10
group by Month
limit 15;
```

```
-> Limit: 15 row(s)  (actual time=4687.406..4687.408 rows=12 loops=1)
    -> Table scan on <temporary>  (actual time=0.003..0.003 rows=12 loops=1)
        -> Aggregate using temporary table  (actual time=4687.405..4687.407 rows=12 loops=1)
            -> Nested loop inner join  (cost=696989.91 rows=635848) (actual time=0.591..4638.984 rows=143391 loops=1)
```

After indexing on month: cost did not decrease. Chose this index to test and see if indexing by flight month would speed up this query.

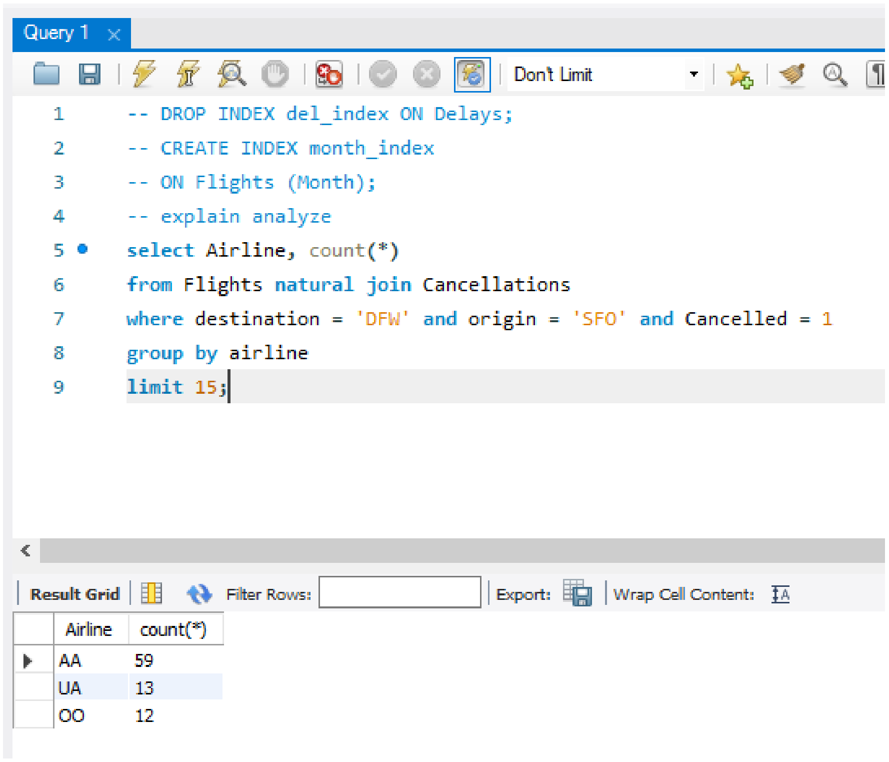


```
-- DROP INDEX del_index ON Delays;
CREATE INDEX month_index
ON Flights (Month);
explain analyze
select count(*)
from Flights natural join Delays
where airline = 'AA' and DepartureDelay < 10
group by Month
limit 15;
```

EXPLAIN:

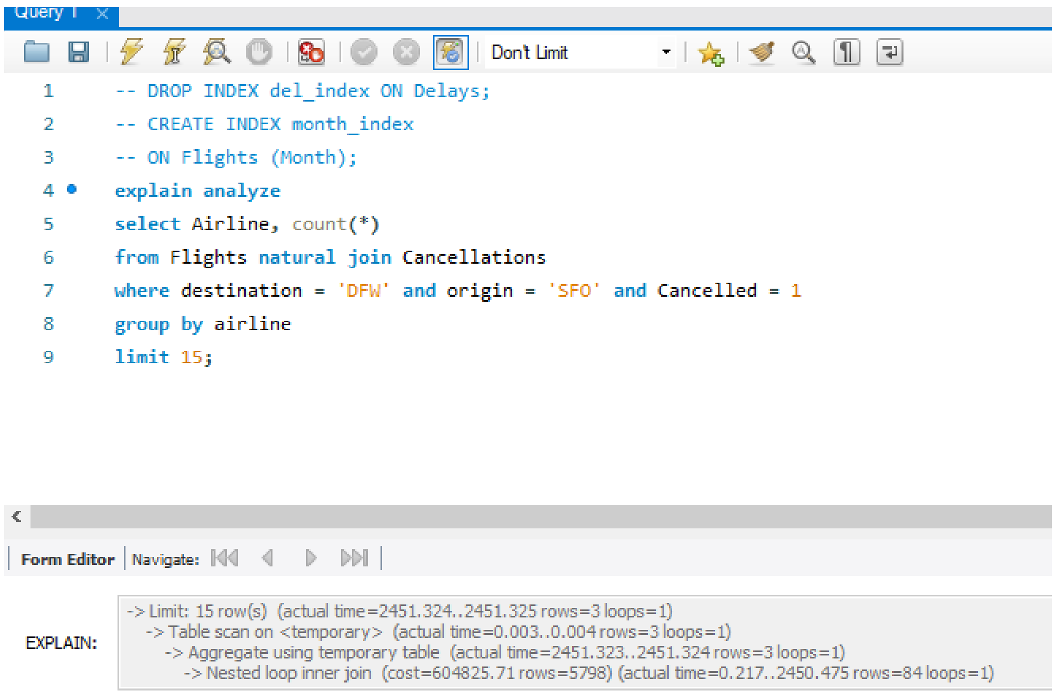
```
-> Limit: 15 row(s)  (actual time=4582.089..4582.091 rows=12 loops=1)
  -> Table scan on <temporary>  (actual time=0.002..0.003 rows=12 loops=1)
    -> Aggregate using temporary table  (actual time=4582.088..4582.090 rows=12 loops=1)
      -> Nested loop inner join  (cost=787472.04 rows=193257) (actual time=0.065..4416.244 rows=576784 loops=1)
```

query 2:



```
-- DROP INDEX del_index ON Delays;
-- CREATE INDEX month_index
-- ON Flights (Month);
-- explain analyze
select Airline, count(*)
from Flights natural join Cancellations
where destination = 'DFW' and origin = 'SFO' and Cancelled = 1
group by airline
limit 15;
```

Airline	count(*)
AA	59
UA	13
OO	12

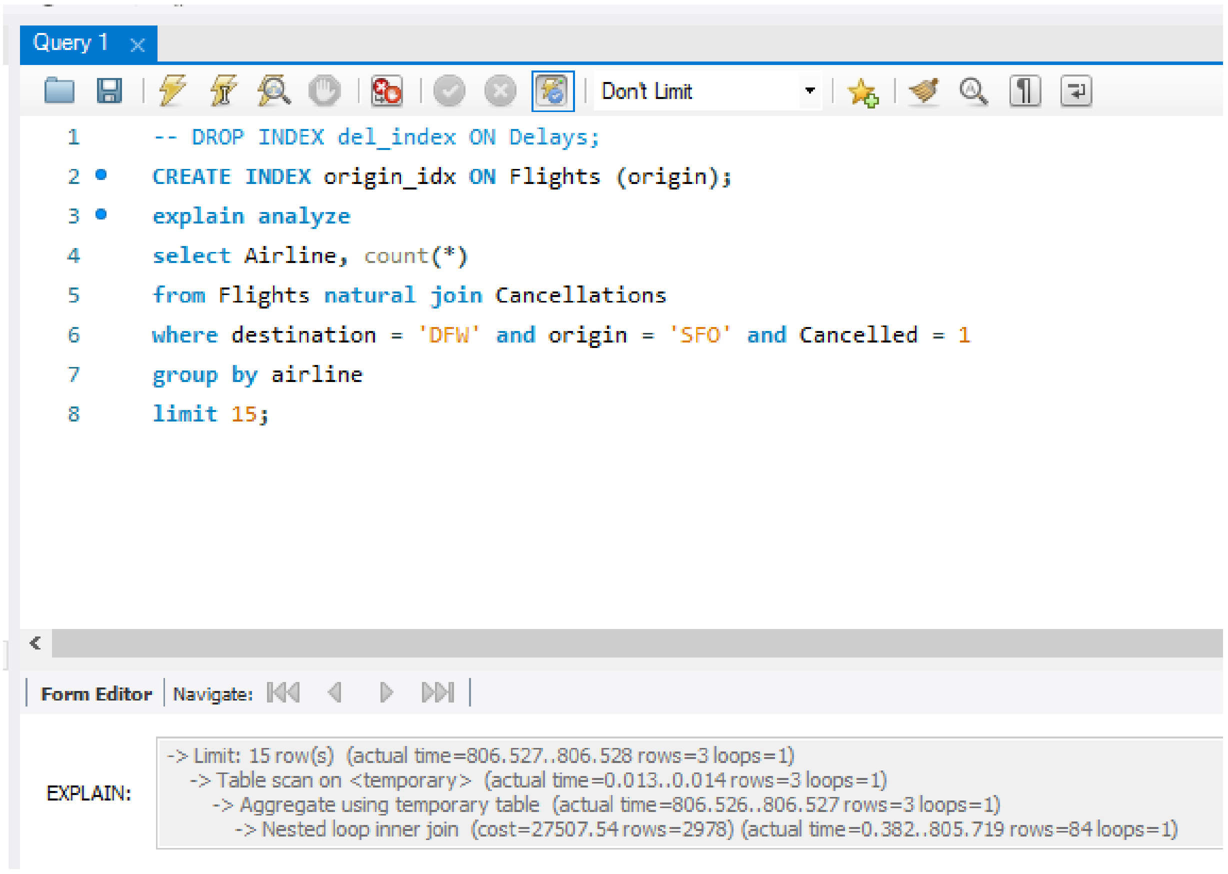


```
-- DROP INDEX del_index ON Delays;
-- CREATE INDEX month_index
-- ON Flights (Month);
explain analyze
select Airline, count(*)
from Flights natural join Cancellations
where destination = 'DFW' and origin = 'SFO' and Cancelled = 1
group by airline
limit 15;
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=2451.324..2451.325 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=0.003..0.004 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=2451.323..2451.324 rows=3 loops=1)
            -> Nested loop inner join  (cost=604825.71 rows=5798) (actual time=0.217..2450.475 rows=84 loops=1)
```

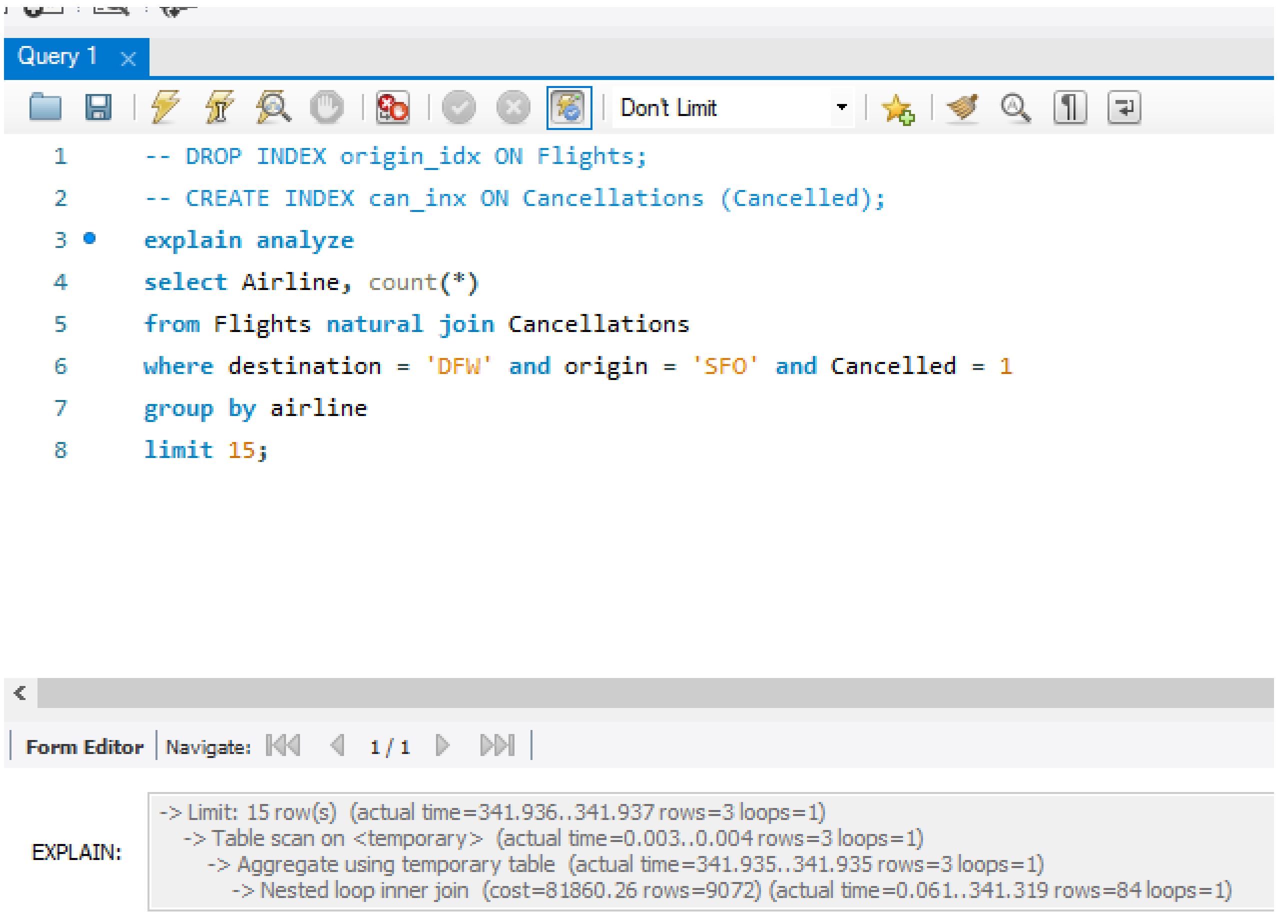
After indexing on Origin name: cost decreased from around 600k to 27k. Chose this index because the query is filtering by origin, so indexing would make it faster for query to filter.



```sql
-- DROP INDEX del_index ON Delays;
CREATE INDEX origin_idx ON Flights (origin);
explain analyze
select Airline, count(*)
from Flights natural join Cancellations
where destination = 'DFW' and origin = 'SFO' and Cancelled = 1
group by airline
limit 15;
```

```
-> Limit: 15 row(s)  (actual time=806.527..806.528 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=0.013..0.014 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=806.526..806.527 rows=3 loops=1)
            -> Nested loop inner join  (cost=27507.54 rows=2978) (actual time=0.382..805.719 rows=84 loops=1)
```

After indexing on canceled (whether or not flight canceled): cost decreased from around 600k to 82k. Chose this index because the query is filtering by cancellation information, so indexing would make it faster for query to filter.

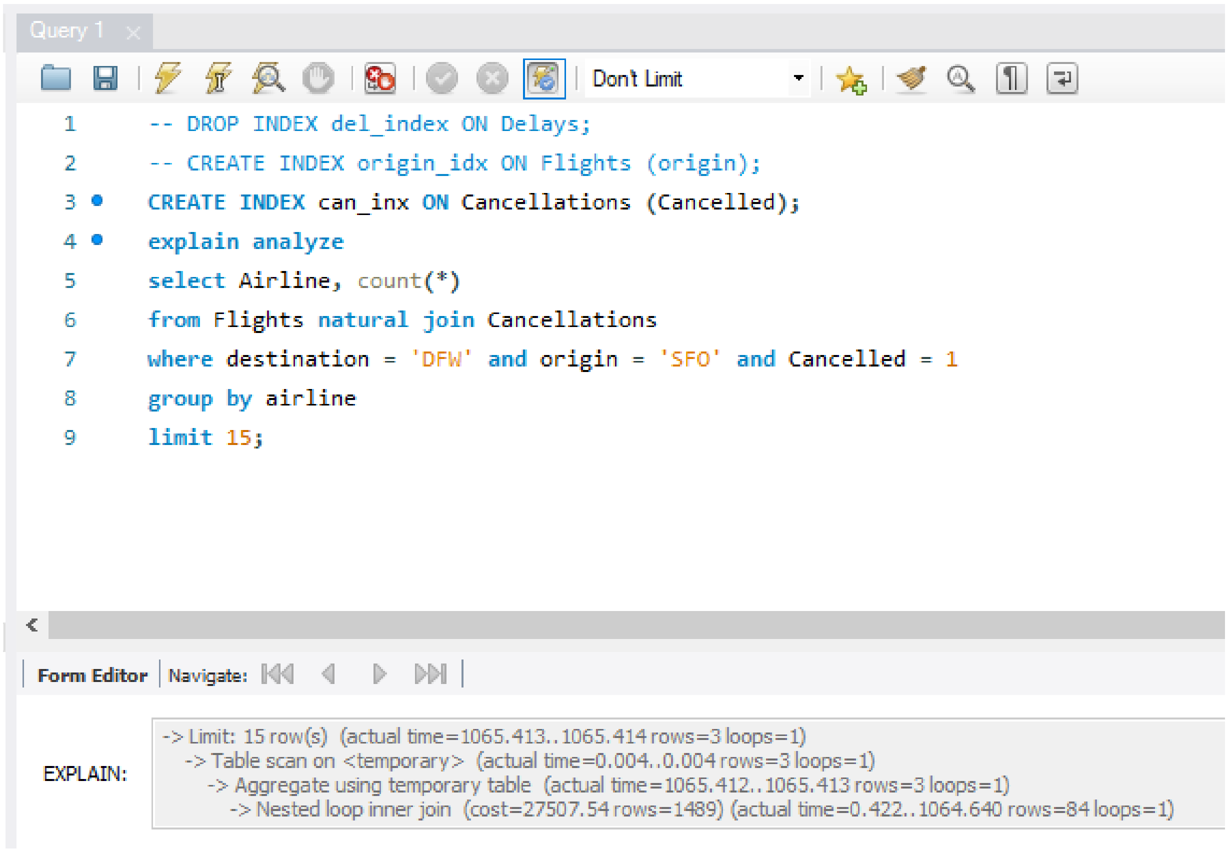


```
-- DROP INDEX origin_idx ON Flights;
-- CREATE INDEX can_inx ON Cancellations (Cancelled);
explain analyze
select Airline, count(*)
from Flights natural join Cancellations
where destination = 'DFW' and origin = 'SFO' and Cancelled = 1
group by airline
limit 15;
```

EXPLAIN:

```
-> Limit: 15 row(s)  (actual time=341.936..341.937 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=0.003..0.004 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=341.935..341.935 rows=3 loops=1)
            -> Nested loop inner join  (cost=81860.26 rows=9072) (actual time=0.061..341.319 rows=84 loops=1)
```

After indexing on both origin name and canceled (whether or not flight canceled): cost did not decrease when compared to only indexing on origin name. We wanted to see if indexing on two important columns would increase the efficiency of the query. We learned that indexing on two columns might not help more than indexing on just one column (at least in this scenario).



```
-- DROP INDEX del_index ON Delays;
-- CREATE INDEX origin_idx ON Flights (origin);
CREATE INDEX can_inx ON Cancellations (Cancelled);
explain analyze
select Airline, count(*)
from Flights natural join Cancellations
where destination = 'DFW' and origin = 'SFO' and Cancelled = 1
group by airline
limit 15;
```

```
-> Limit: 15 row(s)  (actual time=1065.413..1065.414 rows=3 loops=1)
  -> Table scan on <temporary>  (actual time=0.004..0.004 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=1065.412..1065.413 rows=3 loops=1)
      -> Nested loop inner join  (cost=27507.54 rows=1489) (actual time=0.422..1064.640 rows=84 loops=1)
```