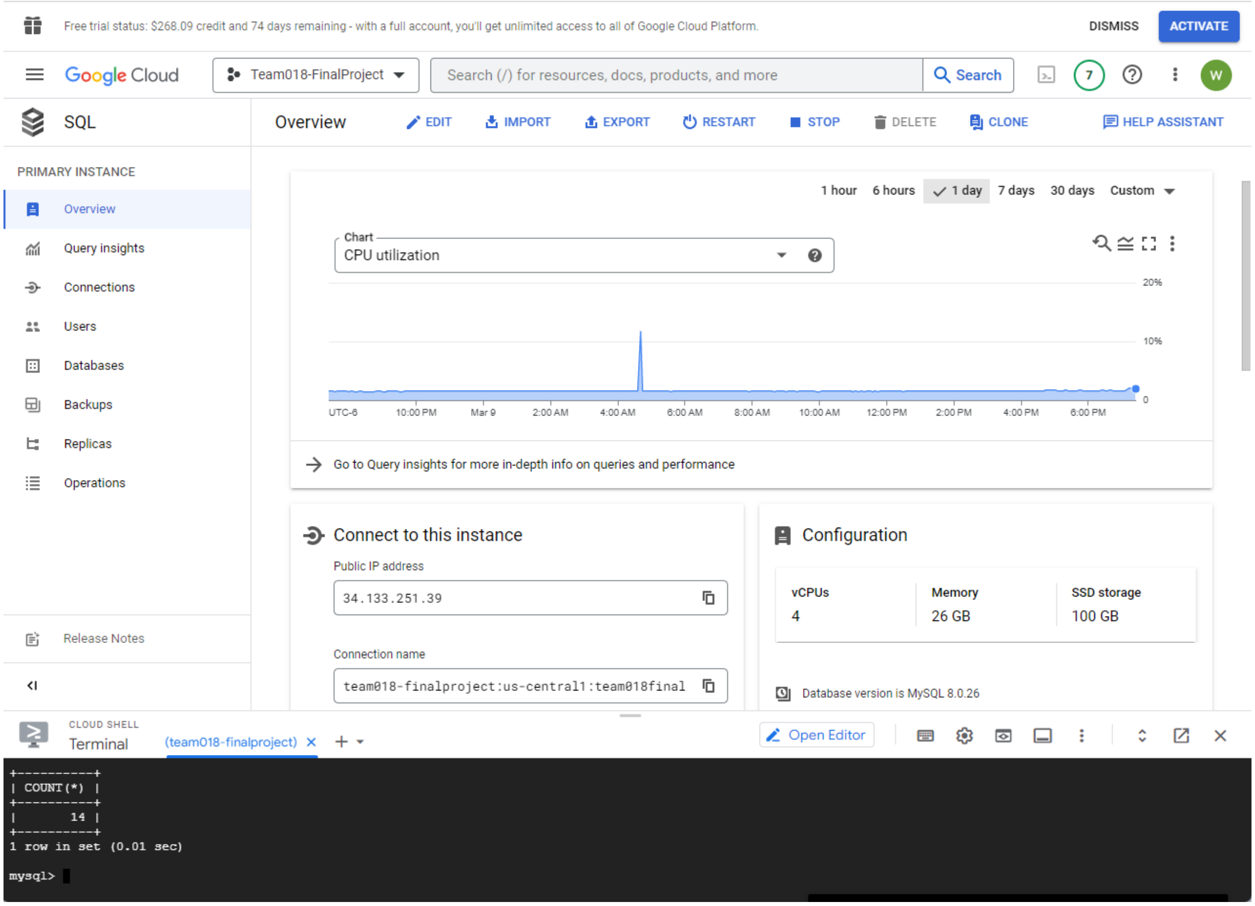
PT 1 Stage 3

Database Implementation and Indexing

Proof of GCP Setup



Free trial status: $268.09 credit and 74 days remaining - with a full account, you'll get unlimited access to all of Google Cloud Platform.

Go to Query insights for more in-depth info on queries and performance

Connect to this instance

Public IP address: 34.133.251.39

Connection name: team018-finalproject:us-central1:team018final

Configuration

vCPUs	Memory	SSD storage
4	26 GB	100 GB

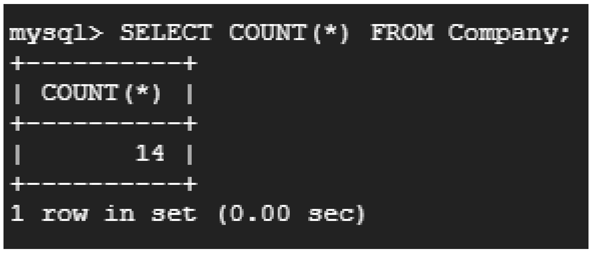
Database version is MySQL 8.0.26

```
+----------+
| COUNT(*) |
+----------+
|       14 |
+----------+
1 row in set (0.01 sec)

mysql> 
```

DDL Commands And Row Counts

Company Table:

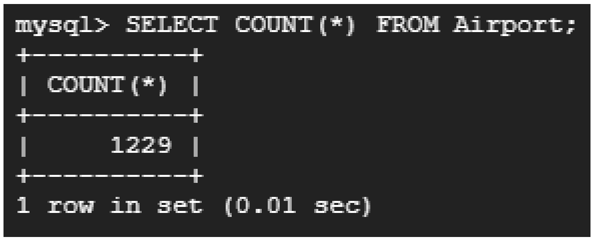
```
CREATE TABLE Company
(
IATA_Designator VARCHAR(2) NOT NULL,
AirlaneName VARCHAR(100),

PRIMARY KEY ( IATA_Designator)
);
```



```
mysql> SELECT COUNT(*) FROM Company;
+----------+
| COUNT(*) |
+----------+
|       14 |
+----------+
1 row in set (0.00 sec)
```

Airport Table

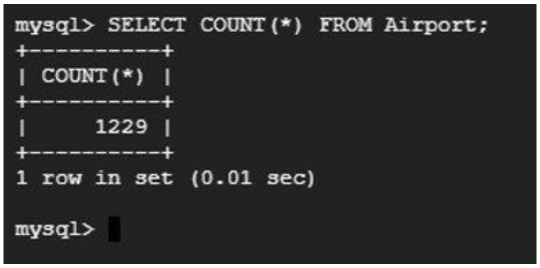
```
CREATE TABLE Airport
(
IATA_Code VARCHAR(100) NOT NULL,
Airport VARCHAR(500),
City VARCHAR(500) NOT NULL,
State VARCHAR(100) NOT NULL,
Latitude DOUBLE,
Longtitude DOUBLE,


PRIMARY KEY (IATA_Code)
);
```



```
mysql> SELECT COUNT(*) FROM Airport;
+----------+
| COUNT(*) |
+----------+
|     1229 |
+----------+
1 row in set (0.01 sec)
```

Flight Table

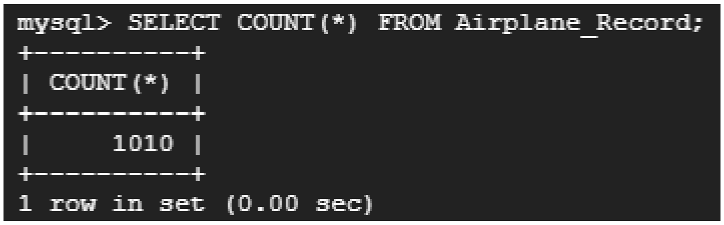
```
CREATE TABLE Flight
(
IATA_Designator VARCHAR(3) NOT NULL,
FlightNumber INT NOT NULL,
Origin_Airport VARCHAR(5) NOT NULL,
Destination_Airport VARCHAR(5) NOT NULL,
Schedule_Departure INT,
Scheduled_Time INT,
Scheduled_Arrival INT,
Distance INT,

PRIMARY KEY (IATA_Code, FlightNumber),
FOREIGN KEY (Origin_Airport) REFERENCES Airport (IATA_Code),
FOREIGN KEY (Destination_Airport) REFERENCES Airport (IATA_Code)
);
```



```
mysql> SELECT COUNT(*) FROM Airport;
+----------+
| COUNT(*) |
+----------+
|     1229 |
+----------+
1 row in set (0.01 sec)

mysql>
```

Airplane Record Table

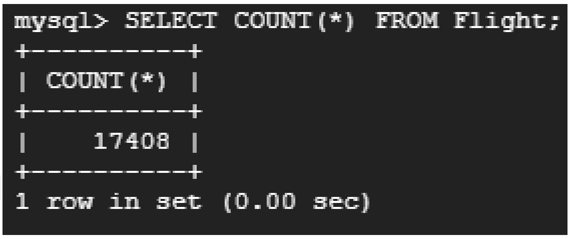
```
CREATE TABLE Airplane_Record
(
FlightNumber INT NOT NULL,
IATA_Designator VARCHAR(2) NOT NULL,
Month VARCHAR(512)  NOT NULL,
Day VARCHAR(512)  NOT NULL,
DayOfWeek VARCHAR(512)  NOT NULL,
TailNumber VARCHAR(100)  NOT NULL,
Departure_Time INT,
Departure_Delay INT,
Taxi_Out INT,
Elapsed_Time INT,
Air_Time INT,
Taxi_In INT,
Arrival_Time INT,
Arrival_Delay INT,

PRIMARY KEY (TailNumber, FlightNumber)
);
```



```
mysql> SELECT COUNT(*) FROM Airplane_Record;
+----------+
| COUNT(*) |
+----------+
|     1010 |
+----------+
1 row in set (0.00 sec)
```

Flight Table

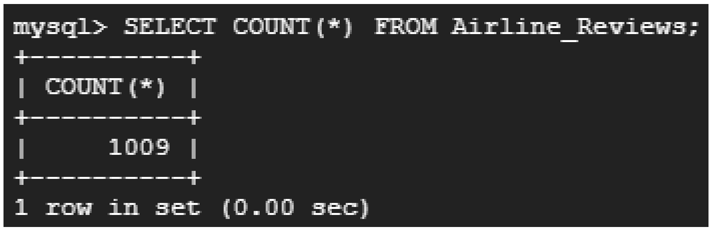
```
CREATE TABLE Flight
(
IATA_Designator VARCHAR(3) NOT NULL,
FlightNumber INT NOT NULL,
Origin_Airport VARCHAR(5) NOT NULL,
Destination_Airport VARCHAR(5) NOT NULL,
Schedule_Departure INT,
Scheduled_Time INT,
Scheduled_Arrival INT,
Distance INT,

PRIMARY KEY (IATA_Code, FlightNumber),
FOREIGN KEY (Origin_Airport) REFERENCES Airport (IATA_Code),
FOREIGN KEY (Destination_Airport) REFERENCES Airport (IATA_Code)
);
```



```
mysql> SELECT COUNT(*) FROM Flight;
+----------+
| COUNT(*) |
+----------+
|    17408 |
+----------+
1 row in set (0.00 sec)
```

Airline Reviews Table

```
CREATE TABLE Airline_Reviews
(
Review_ID INT NOT NULL,
Airline_Name VARCHAR(100) NOT NULL,
Overall INT,
Author VARCHAR(100),
Review_date VARCHAR(100),
Aircraft VARCHAR(100),
Traveller_type VARCHAR(100),
Cabin VARCHAR(100),
Route VARCHAR(100),
Date_flown VARCHAR(100),
Seat_comfort VARCHAR(100),
Cabin_service VARCHAR(100),
Food_bev VARCHAR(100),
Entertainment VARCHAR(100),
Ground_service VARCHAR(100),
Value_for_money VARCHAR(100),
Recommended VARCHAR(100),

PRIMARY KEY (Review_ID)
);
```

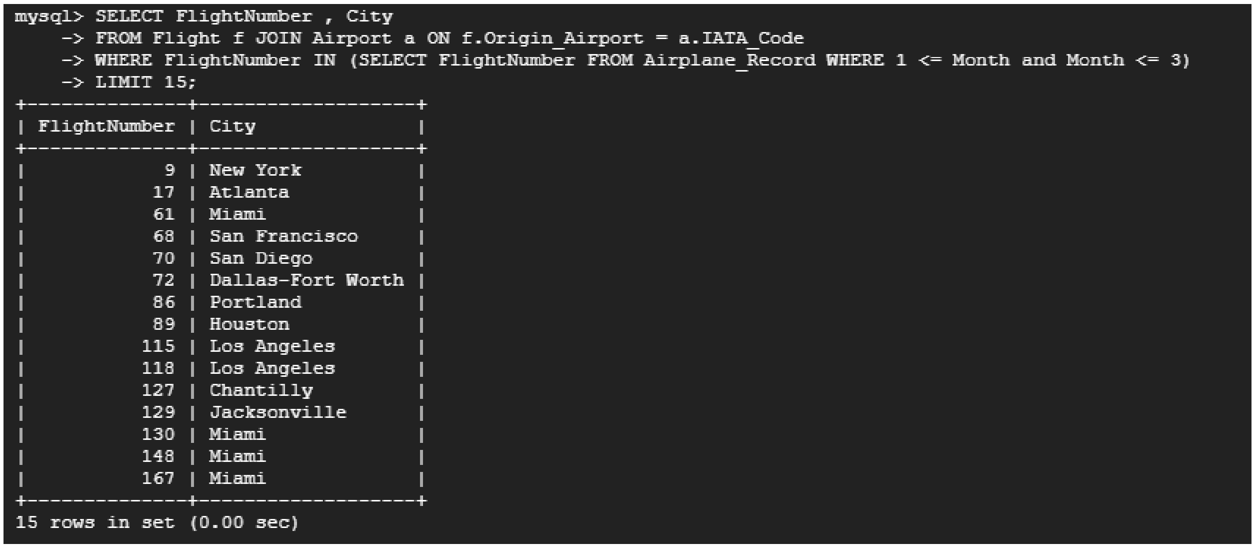


```
mysql> SELECT COUNT(*) FROM Airline_Reviews;
+----------+
| COUNT(*) |
+----------+
|     1009 |
+----------+
1 row in set (0.00 sec)
```

Advanced Queries

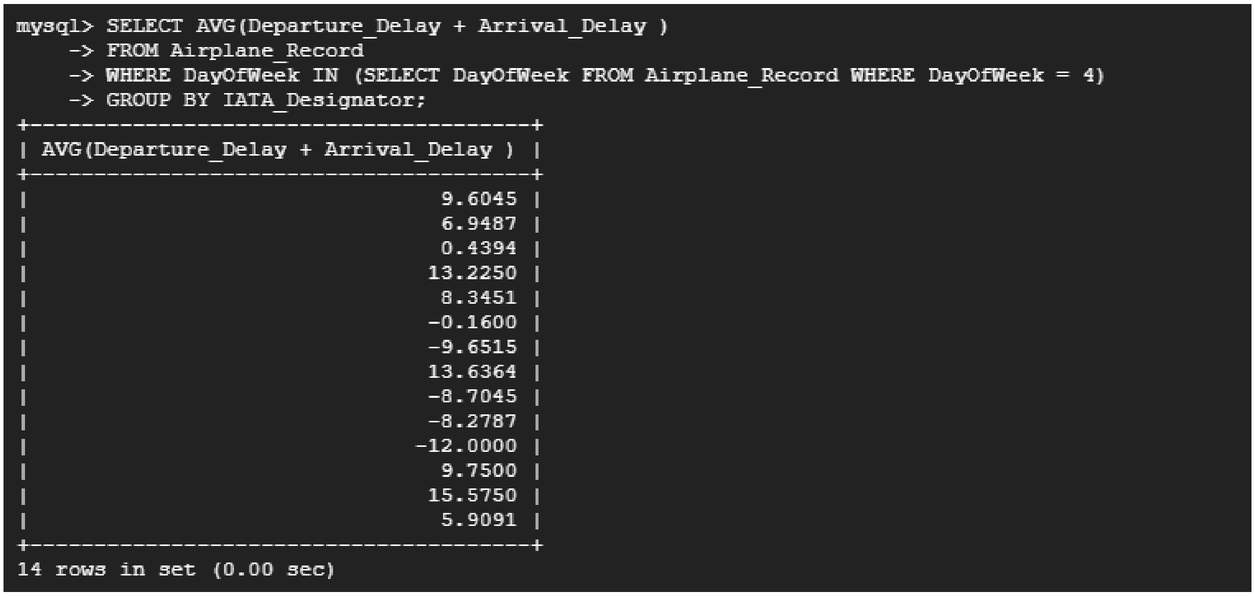
Advanced Query #1:

We are selecting the flight number and city for the flights that were scheduled from the first 3 months of 2015 in this query. (JOIN and SUBQUERY)



```
mysql> SELECT FlightNumber , City
    -> FROM Flight f JOIN Airport a ON f.Origin_Airport = a.IATA_Code
    -> WHERE FlightNumber IN (SELECT FlightNumber FROM Airplane_Record WHERE 1 <= Month and Month <= 3)
    -> LIMIT 15;
+--------------+-------------------+
| FlightNumber | City              |
+--------------+-------------------+
|            9 | New York          |
|           17 | Atlanta           |
|           61 | Miami             |
|           68 | San Francisco     |
|           70 | San Diego         |
|           72 | Dallas-Fort Worth |
|           86 | Portland          |
|           89 | Houston           |
|          115 | Los Angeles       |
|          118 | Los Angeles       |
|          127 | Chantilly         |
|          129 | Jacksonville      |
|          130 | Miami             |
|          148 | Miami             |
|          167 | Miami             |
+--------------+-------------------+
15 rows in set (0.00 sec)
```

Advanced Query #2:



```
mysql> SELECT AVG(Departure_Delay + Arrival_Delay )
    -> FROM Airplane_Record
    -> WHERE DayOfWeek IN (SELECT DayOfWeek FROM Airplane_Record WHERE DayOfWeek = 4)
    -> GROUP BY IATA_Designator;
+--------------------------------------+
| AVG(Departure_Delay + Arrival_Delay ) |
+--------------------------------------+
|                               9.6045 |
|                               6.9487 |
|                               0.4394 |
|                              13.2250 |
|                               8.3451 |
|                              -0.1600 |
|                              -9.6515 |
|                              13.6364 |
|                              -8.7045 |
|                              -8.2787 |
|                             -12.0000 |
|                               9.7500 |
|                              15.5750 |
|                               5.9091 |
+--------------------------------------+
14 rows in set (0.00 sec)
```

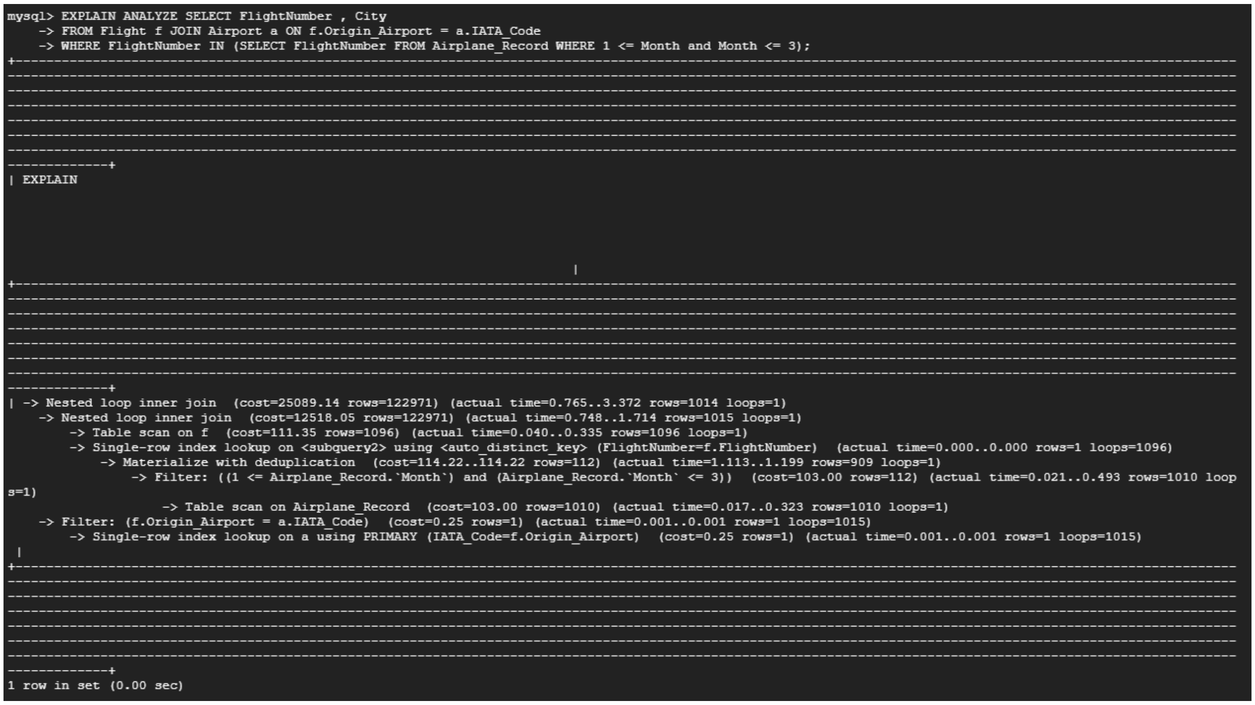
We are computing and selecting the average delay time for flights that were scheduled on Wednesdays in 2015. (GROUP BY and SUBQUERY)

This only has 14 data values because we are using a limited number of rows for our dataset since uploading a larger file doesn't work on GCP.

Indexing

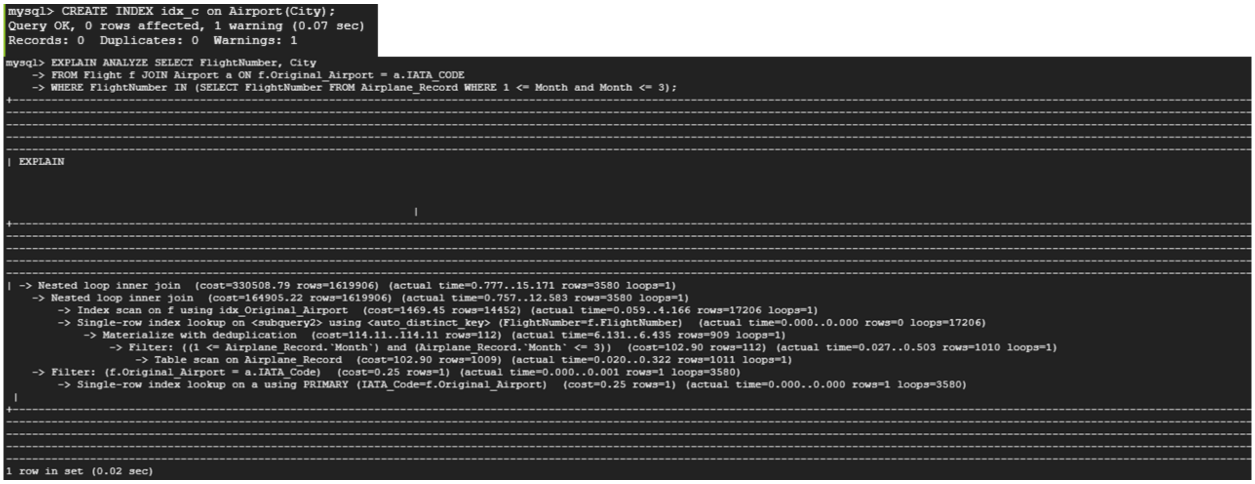
Query 1:

The nested loop inner joins costs are 25089.14 and 12518.05, respectively. These are not very optimal.



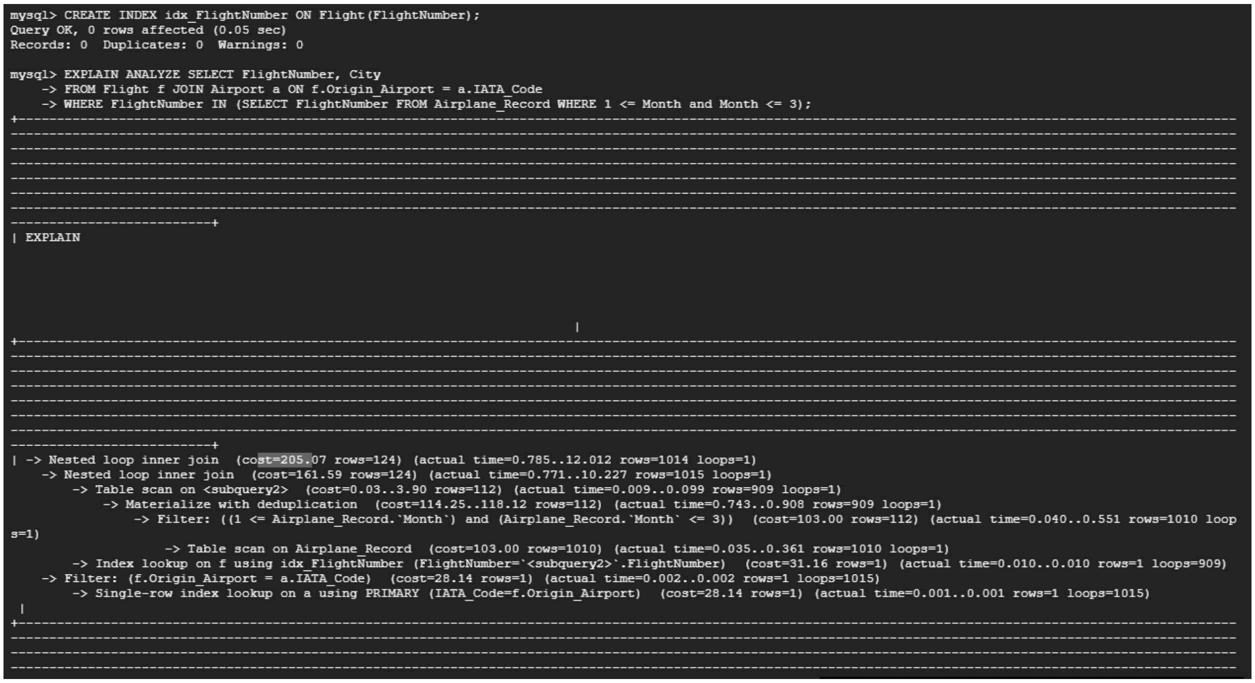
```
mysql> EXPLAIN ANALYZE SELECT FlightNumber , City
    -> FROM Flight f JOIN Airport a ON f.Origin_Airport = a.IATA_Code
    -> WHERE FlightNumber IN (SELECT FlightNumber FROM Airplane_Record WHERE 1 <= Month and Month <= 3);
+--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
------------+
| EXPLAIN




                                                                           |
+--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
------------+
| -> Nested loop inner join  (cost=25089.14 rows=122971) (actual time=0.765..3.372 rows=1014 loops=1)
    -> Nested loop inner join  (cost=12518.05 rows=122971) (actual time=0.748..1.714 rows=1015 loops=1)
        -> Table scan on f  (cost=111.35 rows=1096) (actual time=0.040..0.335 rows=1096 loops=1)
        -> Single-row index lookup on <subquery2> using <auto_distinct_key> (FlightNumber=f.FlightNumber)  (actual time=0.000..0.000 rows=1 loops=1096)
            -> Materialize with deduplication  (cost=114.22..114.22 rows=112) (actual time=1.113..1.199 rows=909 loops=1)
                -> Filter: ((1 <= Airplane_Record.`Month`) and (Airplane_Record.`Month` <= 3))  (cost=103.00 rows=112) (actual time=0.021..0.493 rows=1010 loop
s=1)
                    -> Table scan on Airplane_Record  (cost=103.00 rows=1010) (actual time=0.017..0.323 rows=1010 loops=1)
    -> Filter: (f.Origin_Airport = a.IATA_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1015)
        -> Single-row index lookup on a using PRIMARY (IATA_Code=f.Origin_Airport)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1015)
 |
+--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
------------+
1 row in set (0.00 sec)
```

We added an index using the FlightNumber column because FlightNumbers are unique values for our tables. Adding this index increases the performance of query 1, as seen below:
The nested inner loops now only cost 205.07 and 161.59, respectively, which is a pretty drastic improvement from the original query without indexing.



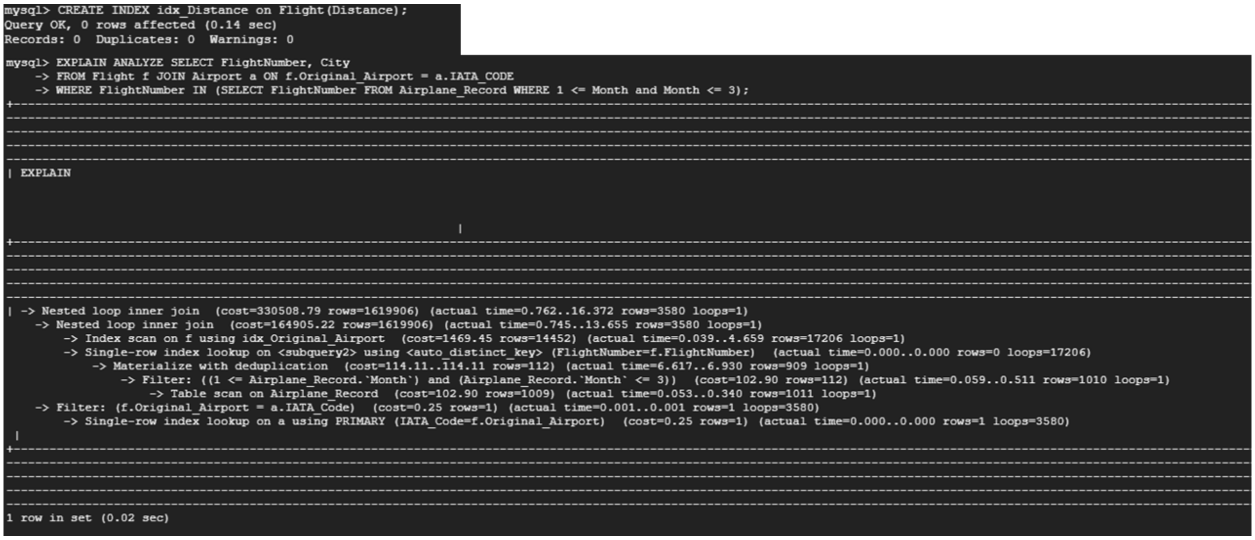
```
mysql> CREATE INDEX idx_c on Airport(City);
Query OK, 0 rows affected, 1 warning (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> EXPLAIN ANALYZE SELECT FlightNumber, City
    -> FROM Flight f JOIN Airport a ON f.Original_Airport = a.IATA_CODE
    -> WHERE FlightNumber IN (SELECT FlightNumber FROM Airplane_Record WHERE 1 <= Month and Month <= 3);
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
| EXPLAIN



                                                                   |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=330508.79 rows=1619906) (actual time=0.777..15.171 rows=3580 loops=1)
    -> Nested loop inner join  (cost=164905.22 rows=1619906) (actual time=0.757..12.583 rows=3580 loops=1)
        -> Index scan on f using idx_Original_Airport  (cost=1469.45 rows=14452) (actual time=0.059..4.166 rows=17206 loops=1)
        -> Single-row index lookup on <subquery2> using <auto_distinct_key> (FlightNumber=f.FlightNumber)  (actual time=0.000..0.000 rows=0 loops=17206)
            -> Materialize with deduplication  (cost=114.11..114.11 rows=112) (actual time=6.131..6.435 rows=909 loops=1)
                -> Filter: ((1 <= Airplane_Record.`Month`) and (Airplane_Record.`Month` <= 3))  (cost=102.90 rows=112) (actual time=0.027..0.503 rows=1010 loops=1)
                    -> Table scan on Airplane_Record  (cost=102.90 rows=1009) (actual time=0.020..0.322 rows=1011 loops=1)
    -> Filter: (f.Original_Airport = a.IATA_Code)  (cost=0.25 rows=1) (actual time=0.000..0.001 rows=1 loops=3580)
        -> Single-row index lookup on a using PRIMARY (IATA_Code=f.Original_Airport)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=3580)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------
1 row in set (0.02 sec)
```

We tried indexing on city for this query, but that index did not provide any positive changes to its performance. This could be because a large number of rows could share the same city attribute, so indexing would still not help improve the efficiency of the query.



```
mysql> CREATE INDEX idx_FlightNumber ON Flight(FlightNumber);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT FlightNumber, City
    -> FROM Flight f JOIN Airport a ON f.Origin_Airport = a.IATA_Code
    -> WHERE FlightNumber IN (SELECT FlightNumber FROM Airplane_Record WHERE 1 <= Month and Month <= 3);
+---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| EXPLAIN




                                                                           |
+---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| -> Nested loop inner join  (cost=205.07 rows=124) (actual time=0.785..12.012 rows=1014 loops=1)
    -> Nested loop inner join  (cost=161.59 rows=124) (actual time=0.771..10.227 rows=1015 loops=1)
        -> Table scan on <subquery2>  (cost=0.03..3.90 rows=112) (actual time=0.009..0.099 rows=909 loops=1)
            -> Materialize with deduplication  (cost=114.25..118.12 rows=112) (actual time=0.743..0.908 rows=909 loops=1)
                -> Filter: ((1 <= Airplane_Record.`Month`) and (Airplane_Record.`Month` <= 3))  (cost=103.00 rows=112) (actual time=0.040..0.551 rows=1010 loop
s=1)
                    -> Table scan on Airplane_Record  (cost=103.00 rows=1010) (actual time=0.035..0.361 rows=1010 loops=1)
        -> Index lookup on f using idx_FlightNumber (FlightNumber=`<subquery2>`.FlightNumber)  (cost=31.16 rows=1) (actual time=0.010..0.010 rows=1 loops=909)
    -> Filter: (f.Origin_Airport = a.IATA_Code)  (cost=28.14 rows=1) (actual time=0.002..0.002 rows=1 loops=1015)
        -> Single-row index lookup on a using PRIMARY (IATA_Code=f.Origin_Airport)  (cost=28.14 rows=1) (actual time=0.001..0.001 rows=1 loops=1015)
 |
+---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------
```

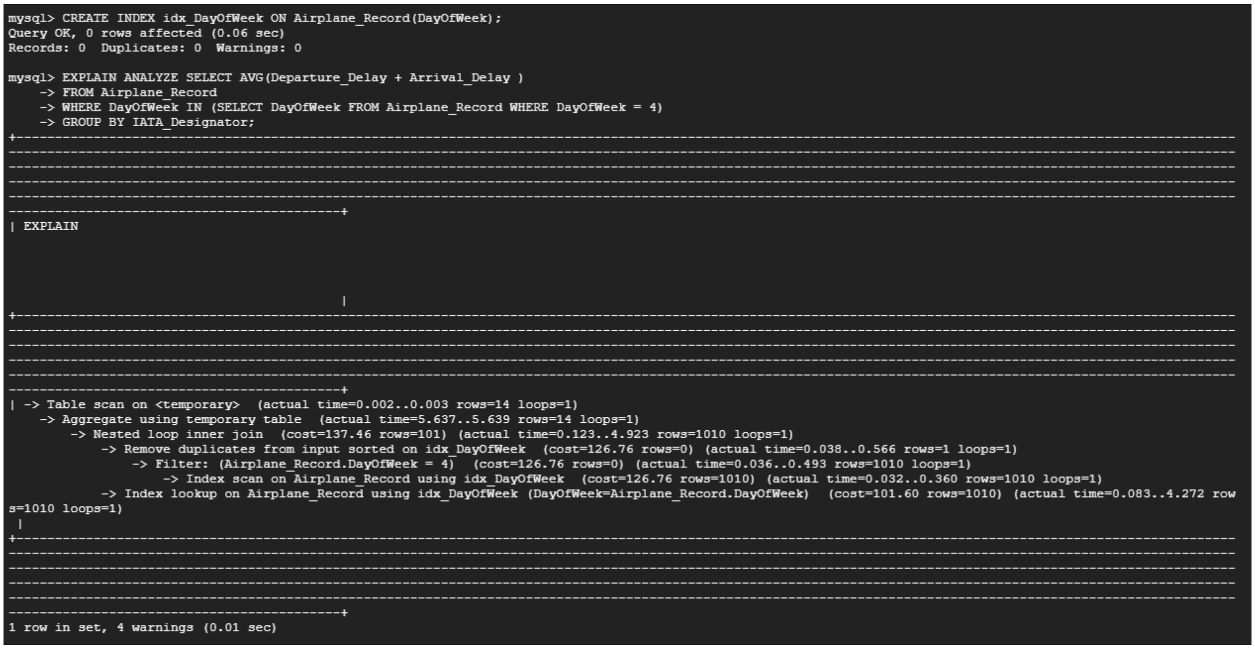
We also tried indexing on distance for this query, and, as seen in the image above, the cost increased greatly to 330508.79 as well (the same cost as indexing on city) and is much greater than the cost when we indexed on flight number. A large number of rows may also share similar distance as flights usually take the same paths during each trip.



```
mysql> CREATE INDEX idx_Distance on Flight(Distance);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT FlightNumber, City
    -> FROM Flight f JOIN Airport a ON f.Original_Airport = a.IATA_CODE
    -> WHERE FlightNumber IN (SELECT FlightNumber FROM Airplane_Record WHERE 1 <= Month and Month <= 3);
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
| EXPLAIN



                                                              |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
| -> Nested loop inner join  (cost=330508.79 rows=1619906) (actual time=0.762..16.372 rows=3580 loops=1)
    -> Nested loop inner join  (cost=164905.22 rows=1619906) (actual time=0.745..13.655 rows=3580 loops=1)
        -> Index scan on f using idx_Original_Airport  (cost=1469.45 rows=14452) (actual time=0.039..4.659 rows=17206 loops=1)
        -> Single-row index lookup on <subquery2> using <auto_distinct_key> (FlightNumber=f.FlightNumber)  (actual time=0.000..0.000 rows=0 loops=17206)
            -> Materialize with deduplication  (cost=114.11..114.11 rows=112) (actual time=6.617..6.930 rows=909 loops=1)
                -> Filter: ((1 <= Airplane_Record.`Month`) and (Airplane_Record.`Month` <= 3))  (cost=102.90 rows=112) (actual time=0.059..0.511 rows=1010 loops=1)
                    -> Table scan on Airplane_Record  (cost=102.90 rows=1009) (actual time=0.053..0.340 rows=1011 loops=1)
    -> Filter: (f.Original_Airport = a.IATA_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3580)
        -> Single-row index lookup on a using PRIMARY (IATA_Code=f.Original_Airport)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=3580)
 |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
1 row in set (0.02 sec)
```

Query 2:

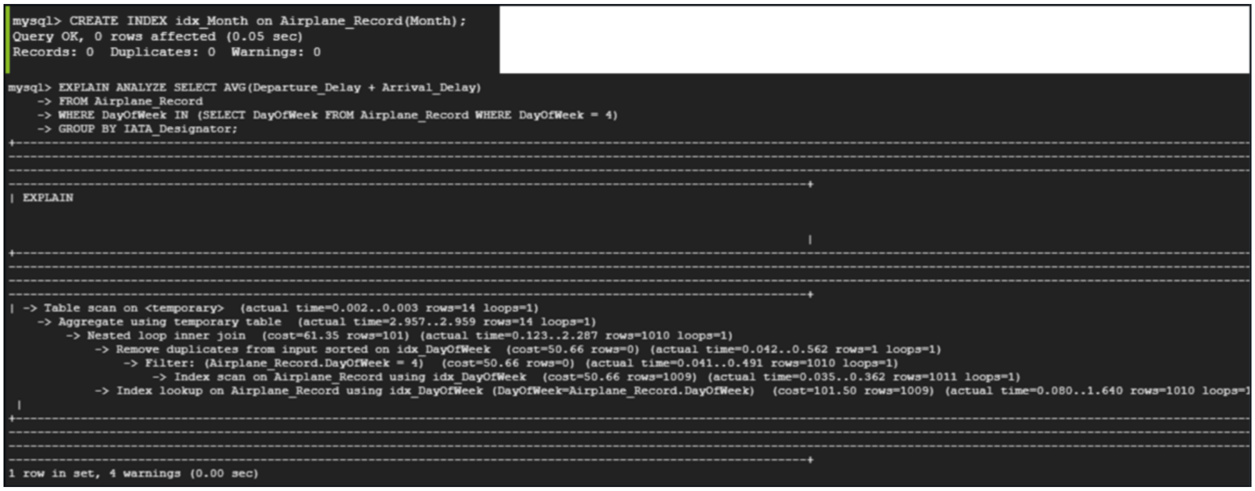
In this query, the nest loop inner join cost is 10405, which is a pretty high number.
We decided to index on the day of the week for this query because we used DayOfWeek in the WHERE clause of the query so indexing by that could possibly help increase the performance of the query.



```
mysql> CREATE INDEX idx_DayOfWeek ON Airplane_Record(DayOfWeek);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT AVG(Departure_Delay + Arrival_Delay )
    -> FROM Airplane_Record
    -> WHERE DayOfWeek IN (SELECT DayOfWeek FROM Airplane_Record WHERE DayOfWeek = 4)
    -> GROUP BY IATA_Designator;
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
-------------------------------------------+
| EXPLAIN



                                           |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
-------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..0.003 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=5.637..5.639 rows=14 loops=1)
        -> Nested loop inner join  (cost=137.46 rows=101) (actual time=0.123..4.923 rows=1010 loops=1)
            -> Remove duplicates from input sorted on idx_DayOfWeek  (cost=126.76 rows=0) (actual time=0.038..0.566 rows=1 loops=1)
                -> Filter: (Airplane_Record.DayOfWeek = 4)  (cost=126.76 rows=0) (actual time=0.036..0.493 rows=1010 loops=1)
                    -> Index scan on Airplane_Record using idx_DayOfWeek  (cost=126.76 rows=1010) (actual time=0.032..0.360 rows=1010 loops=1)
            -> Index lookup on Airplane_Record using idx_DayOfWeek (DayOfWeek=Airplane_Record.DayOfWeek)  (cost=101.60 rows=1010) (actual time=0.083..4.272 row
s=1010 loops=1)
 |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
-------------------------------------------+
1 row in set, 4 warnings (0.01 sec)
```

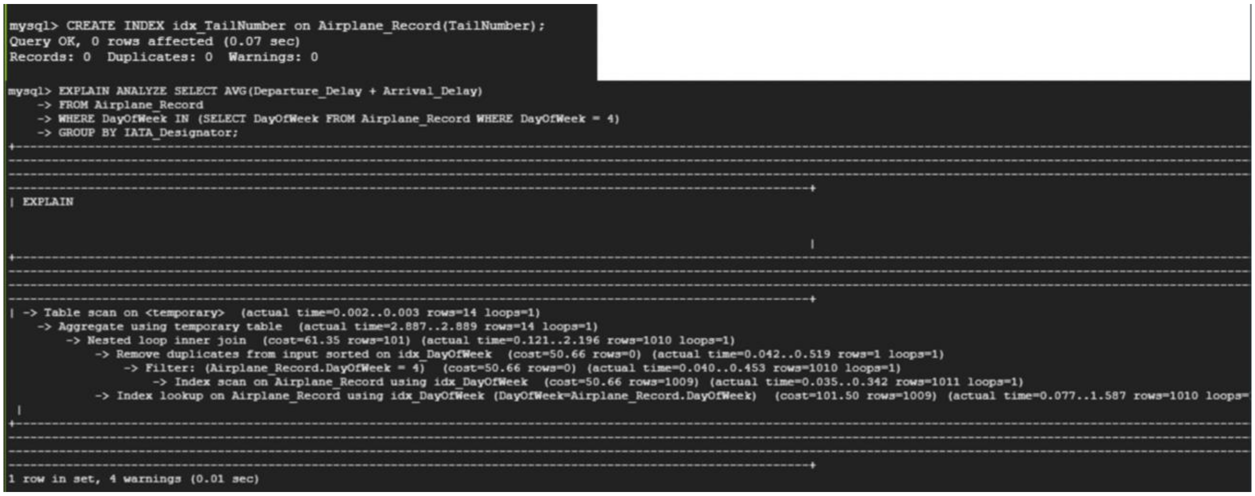
As seen in the EXPLAIN ANALYZE above, indexing by the day of the week did in fact improve the performance of this query. The nest loop inner join cost only 137.46 this time, which is yet another good improvement from the original query.

Next, we tried indexing on the Month. This worked surprisingly well, as the cost was only 61.35 for the inner loop as compared to the 137.46 from the previous indexing cost. This can be seen in the EXPLAIN ANALYZE image below:



```
mysql> CREATE INDEX idx_Month on Airplane_Record(Month);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT AVG(Departure_Delay + Arrival_Delay)
    -> FROM Airplane_Record
    -> WHERE DayOfWeek IN (SELECT DayOfWeek FROM Airplane_Record WHERE DayOfWeek = 4)
    -> GROUP BY IATA_Designator;
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                         |
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..0.003 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=2.957..2.959 rows=14 loops=1)
        -> Nested loop inner join  (cost=61.35 rows=101) (actual time=0.123..2.287 rows=1010 loops=1)
            -> Remove duplicates from input sorted on idx_DayOfWeek  (cost=50.66 rows=0) (actual time=0.042..0.562 rows=1 loops=1)
                -> Filter: (Airplane_Record.DayOfWeek = 4)  (cost=50.66 rows=0) (actual time=0.041..0.491 rows=1010 loops=1)
                    -> Index scan on Airplane_Record using idx_DayOfWeek  (cost=50.66 rows=1009) (actual time=0.035..0.362 rows=1011 loops=1)
            -> Index lookup on Airplane_Record using idx_DayOfWeek (DayOfWeek=Airplane_Record.DayOfWeek)  (cost=101.50 rows=1009) (actual time=0.080..1.640 rows=1010 loops=1
 |
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
1 row in set, 4 warnings (0.00 sec)
```

Finally, we decided to also try indexing on the TailNumber. We decided to do this because it’s a primary key for the Airplane_Record table, so we wanted to see whether it would help. Indexing on TailNumber yielded a cost of 61.35 for next loop inner join, which is the same as the cost when indexed by Month. Thus, we concluded that a good option for our project would be to index by either the TailNumber or Month, based on the context of our functions.



```
mysql> CREATE INDEX idx_TailNumber on Airplane_Record(TailNumber);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT AVG(Departure_Delay + Arrival_Delay)
    -> FROM Airplane_Record
    -> WHERE DayOfWeek IN (SELECT DayOfWeek FROM Airplane_Record WHERE DayOfWeek = 4)
    -> GROUP BY IATA_Designator;
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| EXPLAIN

                                                                                                         |
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.002..0.003 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=2.887..2.889 rows=14 loops=1)
        -> Nested loop inner join  (cost=61.35 rows=101) (actual time=0.121..2.196 rows=1010 loops=1)
            -> Remove duplicates from input sorted on idx_DayOfWeek  (cost=50.66 rows=0) (actual time=0.042..0.519 rows=1 loops=1)
                -> Filter: (Airplane_Record.DayOfWeek = 4)  (cost=50.66 rows=0) (actual time=0.040..0.453 rows=1010 loops=1)
                    -> Index scan on Airplane_Record using idx_DayOfWeek  (cost=50.66 rows=1009) (actual time=0.035..0.342 rows=1011 loops=1)
            -> Index lookup on Airplane_Record using idx_DayOfWeek (DayOfWeek=Airplane_Record.DayOfWeek)  (cost=101.50 rows=1009) (actual time=0.077..1.587 rows=1010 loops=
 |
+---------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
1 row in set, 4 warnings (0.01 sec)
```

Changes Made to Stage 2

- Since we did not have four main entities in our first submission for Stage 2, we found another dataset to join with our current files. It includes the airline reviews for the most popular airlines in 2018 and can be found using this link: https://www.kaggle.com/datasets/efehandanisman/skytrax-airline-reviews?resource=download. Using this, we created our fourth main entity “Reviews” with primary key “airlines” that connects to our company entity. We know that this this data is reliable as it was used for MEF University Big Data Analytics programme for 2018-2019, and it is relevant to our project as it allows us to analyze and connect the raw data behind delays and cancellations and real customer feedback.
- We have also merged and update some of our smaller entities together to better organize our data, and we have remarked the weak entities in our diagram.