

# Part 1

## Database connection on GCP:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to flightdelaypro.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
jackwangcas@cloudshell:~ (flightdelaypro)$ gcloud sql connect flightdatabase --user=root
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 410
Server version: 8.0.31-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>
```

## Data Definition Language (DDL) commands:

```
CREATE TABLE User(  
    user_id INT,  
    username VARCHAR(255),  
    password INT,  
  
    PRIMARY KEY (user_id)  
);
```

```
CREATE TABLE Delay(  
    delay_number INT,  
    user_id INT,  
    flight_number INT,  
    airline_code VARCHAR(2),  
    minutes INT,  
    day_of_week INT,  
    distance INT,  
  
    PRIMARY KEY (delay_number),  
    FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,  
    FOREIGN KEY (flight_number) REFERENCES Operate_Flight(flight_number) ON  
DELETE CASCADE,  
    FOREIGN KEY (airline_code) REFERENCES Airline(airline_code) ON DELETE  
CASCADE  
);
```

```
CREATE TABLE Operate_Flight(  
    flight_number INT,  
    airline_code VARCHAR(2),  
    orig_airport_code VARCHAR(4),  
    dest_airport_code VARCHAR(4),  
  
    PRIMARY KEY (flight_number, airline_code),  
    FOREIGN KEY (airline_code) REFERENCES Airline(airline_code) ON DELETE  
CASCADE,  
    FOREIGN KEY (orig_airport_code) REFERENCES Airport(airport_code) ON  
DELETE CASCADE,  
    FOREIGN KEY (dest_airport_code) REFERENCES Airport(airport_code) ON  
DELETE CASCADE
```

);

```
CREATE TABLE Airport(  
  airport_code VARCHAR(4),  
  airport_name VARCHAR(255),  
  city VARCHAR(255),  
  state VARCHAR(2),
```

```
  PRIMARY KEY (airport_code)  
);
```

```
CREATE TABLE Airline(  
  airline_code VARCHAR(2),  
  airline_name VARCHAR(255),
```

```
  PRIMARY KEY (airline_code)  
);
```

Inserting rows into tables:

```
1 Use flightdatabase;
2
3 SELECT COUNT(*) FROM Operate_Flight
4
5
```

100% 36:3

Result Grid Filter Rows: Search Export:

COUNT(*)
17416

Limit to 1000 rows

```
1 Use flightdatabase;
2
3 SELECT COUNT(*) FROM User;
4
5
```

100% 81:32

Result Grid Filter Rows: Search Export:

COUNT(*)
1077

```
1 Use flightdatabase;
2
3 SELECT COUNT(*) FROM Delay
4
5
```

100% 27:3

Result Grid Filter Rows: Search Export:

COUNT(*)
1048575

# Advanced SQL Queries

## Query 1:



The sql retrieves flights between start and destination cities(Seattle and San Jose in this case). It gives details about the flight, airlines operating them, and the number of delays for those flights.

```
33     Operate_Flight.flight_number,  
34     Operate_Flight.airline_code,  
35     Operate_Flight.orig_airport_code,  
36     Operate_Flight.dest_airport_code,  
37     Airline.airline_name,  
38     COUNT(Delay.delay_number) AS num_delays,  
39     SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations  
40 FROM Operate_Flight  
41 INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code  
42 INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code  
43 INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code  
44 LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number  
45 WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport  
46     AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination  
47     AND Delay.minutes > 30  
48 GROUP BY  
49     Operate_Flight.flight_number,  
50     Operate_Flight.airline_code,  
51     Operate_Flight.orig_airport_code,  
52     Operate_Flight.dest_airport_code,  
53     Airline.airline_name
```

## Results:

```
32 • SELECT  
33     Operate_Flight.flight_number,  
34     Operate_Flight.airline_code,  
35     Operate_Flight.orig_airport_code,  
36     Operate_Flight.dest_airport_code,  
37     Airline.airline_name,  
38     COUNT(Delay.delay_number) AS num_delays,  
39     SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations  
40 FROM Operate_Flight  
41 INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code  
42 INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code  
43 INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code  
44 LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number  
45 WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport  
46     AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination  
47     AND Delay.minutes > 30  
48 GROUP BY  
49     Operate_Flight.flight_number,  
50     Operate_Flight.airline_code,
```

26:67

Result Grid  Filter Rows:  Export: 

flight_num...	airline_code	orig_airport_co...	dest_airport_co...	airline_name	num_delays	num_cancellatio...	
3858	WN	SEA	SJC	Southwest Airlines Co.	3	3	
4802	OO	SEA	SJC	Skywest Airlines Inc.	6	6	
4733	OO	SEA	SJC	Skywest Airlines Inc.	14	14	
4671	WN	SEA	SJC	Southwest Airlines Co.	22	22	
895	WN	SEA	SJC	Southwest Airlines Co.	32	32	
200	AS	SEA	SJC	Alaska Airlines Inc.	37	37	
326	AS	SEA	SJC	Alaska Airlines Inc.	37	37	
330	AS	SEA	SJC	Alaska Airlines Inc.	38	38	
4727	OO	SEA	SJC	Skywest Airlines Inc.	38	38	
324	AS	SEA	SJC	Alaska Airlines Inc.	49	49	
334	AS	SEA	SJC	Alaska Airlines Inc.	59	59	
322	AS	SEA	SJC	Alaska Airlines Inc.	60	60	
338	AS	SEA	SJC	Alaska Airlines Inc.	60	60	
336	AS	SEA	SJC	Alaska Airlines Inc.	85	85	
328	AS	SEA	SJC	Alaska Airlines Inc.	91	91	

### Query 2:

Retrieves information about airlines, including the airline code, the name of the airline, the total number of records in our database that airline has, and the percentage of those records that are delayed (positive numbers).

```

25     a.airline_code,
26     a.airline_name,
27     COUNT(d.delay_number) AS total_records,
28     ROUND((SUM(CASE WHEN d.minutes > 0 THEN 1 ELSE 0 END) / COUNT(d.delay_number)) * 100, 2) AS percentage_with_delay
29 FROM
30     Airline a
31 LEFT JOIN
32     Delay d ON a.airline_code = d.airline_code
33 WHERE
34     a.airline_code = 'UA'
35     AND d.day_of_week = 1
36     AND d.distance BETWEEN 1000 AND 2000
37 GROUP BY

```

Result:

```
24 • SELECT
25     a.airline_code,
26     a.airline_name,
27     COUNT(d.delay_number) AS total_records,
28     ROUND((SUM(CASE WHEN d.minutes > 0 THEN 1 ELSE 0 END) / COUNT(d.delay_number)) * 100, 2) AS percentage_with_delay
29 FROM
30     Airline a
31 LEFT JOIN
32     Delay d ON a.airline_code = d.airline_code
33 WHERE
34     a.airline_code = 'UA'
35     AND d.day_of_week = 1
36     AND d.distance BETWEEN 1000 AND 2000
37 GROUP BY
38     a.airline_code, a.airline_name;
39
40
```

11:53

Result Grid

Filter Rows:

Search

Export:

airline_code	airline_name	total_recor...	percentage_with_del...
UA	United Air Lines Inc.	5040	44.19

# Part 2

## Query 1

Query 1 before any indexing:

```
'-> Sort: num_delays, num_cancellations (actual time=41.731..41.732 rows=15 loops=1)
-> Table scan on <temporary> (actual time=41.693..41.700 rows=15 loops=1)
-> Aggregate using temporary table (actual time=41.690..41.690 rows=15 loops=1)
-> Nested loop inner join (cost=10550.33 rows=8348) (actual time=2.314..32.076 rows=4832 loops=1)
-> Nested loop inner join (cost=1784.17 rows=167) (actual time=1.098..2.060 rows=15 loops=1)
-> Nested loop inner join (cost=1199.60 rows=1670) (actual time=1.070..1.675 rows=127 loops=1)
-> Nested loop inner join (cost=615.02 rows=1670) (actual time=1.057..1.444 rows=127 loops=1)
-> Filter: (destination_airport.city = "San Jose") (cost=30.45 rows=30) (actual time=0.809..0.841 rows=1 loops=1)
-> Table scan on destination_airport (cost=30.45 rows=297) (actual time=0.689..0.799 rows=322 loops=1)
-> Filter: (Operate_Flight.orig_airport_code is not null) (cost=14.25 rows=56) (actual time=0.247..0.590 rows=127 loops=1)
-> Index lookup on Operate_Flight using dest_airport_code (dest_airport_code=destination_airport.airport_code) (cost=14.25 rows=56) (actual time=0.244..0.563 rows=127 loops=1)
-> Single-row index lookup on Airline using PRIMARY (airline_code=Operate_Flight.airline_code) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=127)
-> Filter: (start_airport.city = "Seattle") (cost=0.25 rows=0.1) (actual time=0.003..0.003 rows=0 loops=127)
-> Single-row index lookup on start_airport using PRIMARY (airport_code=Operate_Flight.orig_airport_code) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=127)
-> Filter: (Delay.minutes > 30) (cost=37.52 rows=50) (actual time=0.559..1.973 rows=322 loops=15)
-> Index lookup on Delay using fk_flight_number (flight_number=Operate_Flight.flight_number) (cost=37.52 rows=150) (actual time=0.557..1.933 rows=365 loops=15)
```

Indexing design 1: add Airport.city as index

```
'-> Sort: num_delays, num_cancellations (actual time=36.962..36.964 rows=15 loops=1)
-> Table scan on <temporary> (actual time=36.912..36.920 rows=15 loops=1)
-> Aggregate using temporary table (actual time=36.908..36.908 rows=15 loops=1)
-> Nested loop inner join (cost=30.42 rows=9) (actual time=0.953..29.055 rows=4832 loops=1)
-> Nested loop inner join (cost=20.45 rows=0.2) (actual time=0.254..0.694 rows=15 loops=1)
-> Nested loop inner join (cost=20.38 rows=0.2) (actual time=0.243..0.601 rows=15 loops=1)
-> Nested loop inner join (cost=0.70 rows=1) (actual time=0.015..0.025 rows=1 loops=1)
-> Covering index lookup on start_airport using city_idx (city="Seattle") (cost=0.35 rows=1) (actual time=0.011..0.014 rows=1 loops=1)
-> Covering index lookup on destination_airport using city_idx (city="San Jose") (cost=0.35 rows=1) (actual time=0.003..0.010 rows=1 loops=1)
-> Filter: (Operate_Flight.orig_airport_code = start_airport.airport_code) (cost=14.08 rows=0.2) (actual time=0.227..0.572 rows=15 loops=1)
-> Index lookup on Operate_Flight using dest_airport_code (dest_airport_code=destination_airport.airport_code) (cost=14.08 rows=56) (actual time=0.222..0.504 rows=127 loops=1)
-> Single-row index lookup on Airline using PRIMARY (airline_code=Operate_Flight.airline_code) (cost=0.78 rows=1) (actual time=0.005..0.005 rows=1 loops=15)
-> Filter: (Delay.minutes > 30) (cost=63.80 rows=50) (actual time=0.499..1.863 rows=322 loops=15)
-> Index lookup on Delay using fk_flight_number (flight_number=Operate_Flight.flight_number) (cost=63.80 rows=150) (actual time=0.498..1.824 rows=365 loops=15)
```

Indexing design 2: add Delay.minutes as index

```
'-> Sort: num_delays, num_cancellations (actual time=20.204..20.206 rows=15 loops=1)
-> Table scan on <temporary> (actual time=20.163..20.169 rows=15 loops=1)
-> Aggregate using temporary table (actual time=20.160..20.160 rows=15 loops=1)
-> Nested loop inner join (cost=10550.33 rows=6476) (actual time=0.959..18.995 rows=631 loops=1)
-> Nested loop inner join (cost=1784.17 rows=167) (actual time=0.471..1.121 rows=15 loops=1)
-> Nested loop inner join (cost=1199.60 rows=1670) (actual time=0.446..0.856 rows=127 loops=1)
-> Nested loop inner join (cost=615.02 rows=1670) (actual time=0.437..0.693 rows=127 loops=1)
-> Filter: (destination_airport.city = "San Jose") (cost=30.45 rows=30) (actual time=0.227..0.258 rows=1 loops=1)
-> Table scan on destination_airport (cost=30.45 rows=297) (actual time=0.107..0.213 rows=322 loops=1)
-> Filter: (Operate_Flight.orig_airport_code is not null) (cost=14.25 rows=56) (actual time=0.209..0.426 rows=127 loops=1)
-> Index lookup on Operate_Flight using dest_airport_code (dest_airport_code=destination_airport.airport_code) (cost=14.25 rows=56) (actual time=0.207..0.410 rows=127 loops=1)
```

- > Single-row index lookup on Airline using PRIMARY (airline\_code=Operate\_Flight.airline\_code) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=127)
- > Filter: (start\_airport.city = "Seattle") (cost=0.25 rows=0.1) (actual time=0.002..0.002 rows=0 loops=127)
- > Single-row index lookup on start\_airport using PRIMARY (airport\_code=Operate\_Flight.orig\_airport\_code) (cost=0.25 rows=1) (actual time=0.001..0.002 rows=1 loops=127)
- > Filter: (Delay.minutes > 30) (cost=37.51 rows=39) (actual time=0.311..1.187 rows=42 loops=15)
- > Index lookup on Delay using fk\_flight\_number (flight\_number=Operate\_Flight.flight\_number) (cost=37.51 rows=150) (actual time=0.309..1.164 rows=365 loops=15)

### Indexing design 3: add Airline.airline\_name as index

- '-> Sort: num\_delays, num\_cancellations (actual time=15.458..15.459 rows=15 loops=1)
- > Table scan on <temporary> (actual time=15.417..15.422 rows=15 loops=1)
- > Aggregate using temporary table (actual time=15.413..15.413 rows=15 loops=1)
- > Nested loop inner join (cost=10550.33 rows=8348) (actual time=0.715..14.291 rows=631 loops=1)
  - > Nested loop inner join (cost=1784.17 rows=167) (actual time=0.376..0.998 rows=15 loops=1)
  - > Nested loop inner join (cost=1199.60 rows=1670) (actual time=0.353..0.739 rows=127 loops=1)
    - > Nested loop inner join (cost=615.02 rows=1670) (actual time=0.345..0.590 rows=127 loops=1)
      - > Filter: (destination\_airport.city = "San Jose") (cost=30.45 rows=30) (actual time=0.167..0.197 rows=1 loops=1)
      - > Table scan on destination\_airport (cost=30.45 rows=297) (actual time=0.048..0.152 rows=322 loops=1)
      - > Filter: (Operate\_Flight.orig\_airport\_code is not null) (cost=14.25 rows=56) (actual time=0.176..0.383 rows=127 loops=1)
      - > Index lookup on Operate\_Flight using dest\_airport\_code (dest\_airport\_code=destination\_airport.airport\_code) (cost=14.25 rows=56) (actual time=0.175..0.369 rows=127 loops=1)
    - > Single-row index lookup on Airline using PRIMARY (airline\_code=Operate\_Flight.airline\_code) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=127)
    - > Filter: (start\_airport.city = "Seattle") (cost=0.25 rows=0.1) (actual time=0.002..0.002 rows=0 loops=127)
  - > Single-row index lookup on start\_airport using PRIMARY (airport\_code=Operate\_Flight.orig\_airport\_code) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=127)
  - > Filter: (Delay.minutes > 30) (cost=37.52 rows=50) (actual time=0.229..0.882 rows=42 loops=15)
  - > Index lookup on Delay using fk\_flight\_number (flight\_number=Operate\_Flight.flight\_number) (cost=37.52 rows=150) (actual time=0.228..0.861 rows=365 loops=15)

We notice that indexing on Airport.city significantly improved the cost of our query the most. The cost of nested loop inner joins went from 10550.33 to only 30.42, and all the other nested loop joins went from thousands/hundreds to 2 digit numbers. We believe this improved the cost because we use Airport.city in the WHERE clause, and there are a lot of unique values in Airport.city. Adding an index on Airport.city will help finding the specified city a lot faster.

Even though Delay.minutes is also used in the WHERE clause, there are too many values of Delay.minutes that are greater than 30, so the indexing wouldn't give too much improvement. We only can see that some actual time got halved, and the number of rows when filtering Delay.minutes reduced by %.

In the case of adding Airline.airline\_name as index, it seems that adding this index only reduced the number of rows that the query needed to process, and the actual time for some parts as well. It didn't improve the cost because it is only used in the GROUP BY clause, and there isn't a lot of unique airline\_name, so indexing on it wouldn't help too much with cost.



## Query 2

Query 2 before indexing:

```
'-> Group aggregate: count(d.delay_number), sum((case when (d.minutes > 0) then 1 else 0 end)),
count(d.delay_number) (cost=2774.12 rows=1732) (actual time=155.017..155.018 rows=1 loops=1)
  -> Filter: ((d.day_of_week = 1) and (d.distance between 1000 and 2000)) (cost=2600.94 rows=1732) (actual
time=9.722..153.980 rows=5040 loops=1)
    -> Index lookup on d using airline_code (airline_code="UA") (cost=2600.94 rows=155884) (actual
time=0.229..147.589 rows=87606 loops=1)
,
```

Indexing design 1: add Delay.distance as index

```
'-> Group aggregate: count(d.delay_number), sum((case when (d.minutes > 0) then 1 else 0 end)),
count(d.delay_number) (cost=3819.75 rows=6960) (actual time=155.064..155.065 rows=1 loops=1)
  -> Filter: ((d.day_of_week = 1) and (d.distance between 1000 and 2000)) (cost=3123.75 rows=6960) (actual
time=10.056..154.105 rows=5040 loops=1)
    -> Index lookup on d using airline_code (airline_code="UA") (cost=3123.75 rows=155884) (actual
time=0.308..147.219 rows=87606 loops=1)
,
```

Indexing design 2: add Delay.day\_of\_week as index

```
'-> Group aggregate: count(d.delay_number), sum((case when (d.minutes > 0) then 1 else 0 end)),
count(d.delay_number) (cost=3560.65 rows=5664) (actual time=161.490..161.491 rows=1 loops=1)
  -> Filter: ((d.day_of_week = 1) and (d.distance between 1000 and 2000)) (cost=2994.20 rows=5664) (actual
time=10.078..160.463 rows=5040 loops=1)
    -> Index lookup on d using airline_code (airline_code="UA") (cost=2994.20 rows=155884) (actual
time=0.195..153.774 rows=87606 loops=1)
,
```

Indexing design 3: add Airline.airline\_name as index

```
'-> Group aggregate: count(d.delay_number), sum((case when (d.minutes > 0) then 1 else 0 end)),
count(d.delay_number) (cost=2774.12 rows=1732) (actual time=162.625..162.626 rows=1 loops=1)
  -> Filter: ((d.day_of_week = 1) and (d.distance between 1000 and 2000)) (cost=2600.94 rows=1732) (actual
time=11.852..161.620 rows=5040 loops=1)
    -> Index lookup on d using airline_code (airline_code="UA") (cost=2600.94 rows=155884) (actual
time=0.188..154.714 rows=87606 loops=1)
,
```

We can see that adding any of the relevant indexes wouldn't improve any cost, but rather increased the cost. This is because none of them are "unique" enough for indexing to improve the performance. Rather, adding useless indices will add some extra costs.

For example, we want to find Delay.distance between 1000 to 2000. However, most of the data in our database has a distance from 1000 to 2000. So indexing it wouldn't help making the searching faster.

As for using day\_of\_week as an index, since there are only 7 days in a week, indexing using day\_of\_week won't make much of a difference in performance as they are all very similar from one another. We can see that in the EXPLAIN ANALYSIS of design 2. The cost was not lowered at all and was even increased a little bit due to the inherent cost of adding a new index.

It is essentially the same for using `airline_name` as index since there are only a few airline companies in the U.S. Indexing by this attribute won't make the query any faster either.